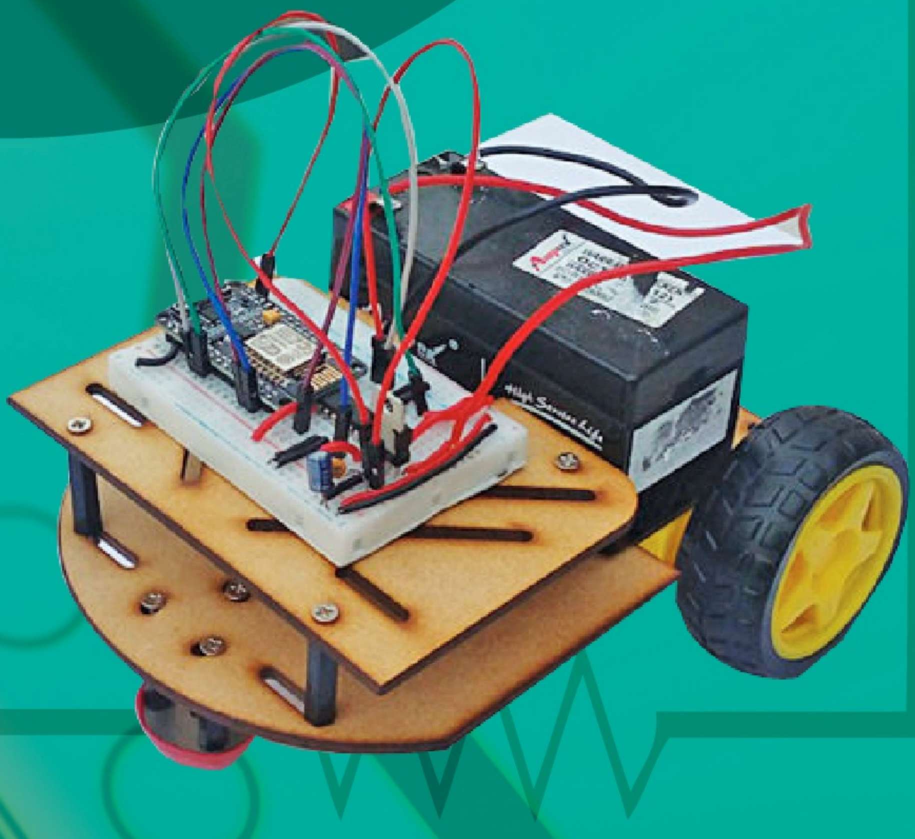
Price: ₹500

electronics FOR YOU
YOURS SINCE 1969

TOP 50 DIY PROJECTS



Contents

Projects Page

Projects Page

1

DESIGN YOUR OWN LOW-COST IOT ROBOT

BY PAMARTHI KANAKARAJA

There are many types of robots, from simple toy cars to advanced industrial robots. The Wi-Fi-controlled IoT robot described here uses NodeMCU and Blynk app. It can be controlled wirelessly using any Wi-Fi enabled Android smartphone. The prototype wired on the breadboard is shown in Fig. 1.

Circuit and working

The block diagram of the low-cost Internet of Things (IoT) robot is shown in Fig. 2, and its circuit diagram is shown in Fig. 3. It is built around NodeMCU (board1), 5V regulator 7805 (IC1), motor driver (IC2), two DC motors (M1 and M2), 12V battery and a few other components.

The project programming of NodeMCU and interfacing Blynk app on Android device.

NodeMCU

NodeMCU is the heart of this project. Pin details of NodeMCU module are shown in Fig. 4. It is a small board compatible with breadboards and has an ESP8266 Wi-Fi controller chip operating at 2.4GHz frequency. It has micro USB port for power, programming and debugging. It also has reset and flash buttons. It can be powered using 5V via micro USB port.

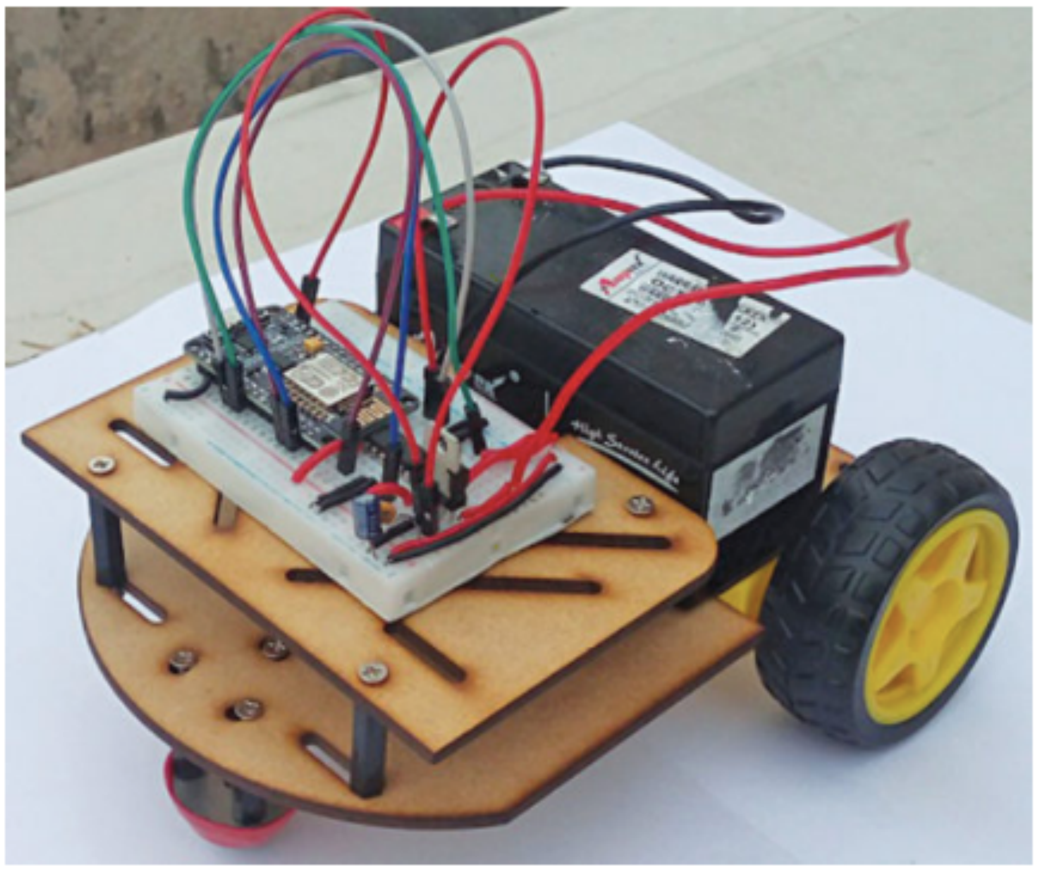


Fig. 1: IoT robot's prototype built at EFY Lab

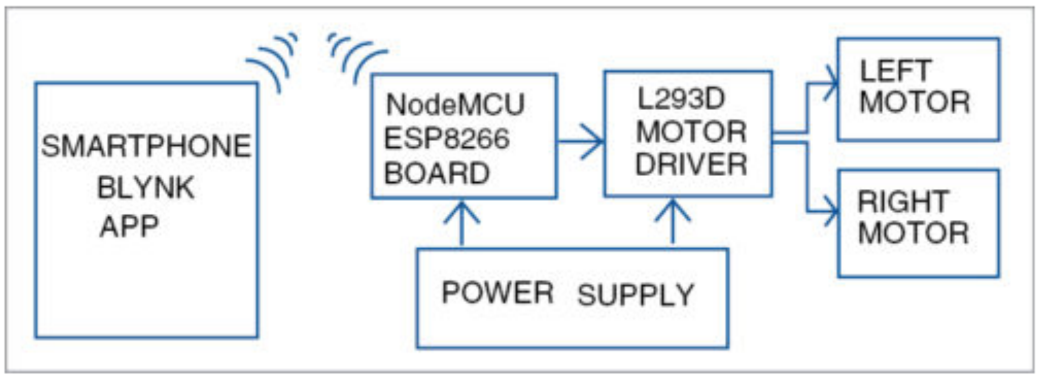


Fig. 2: Block diagram of the low-cost IoT robot

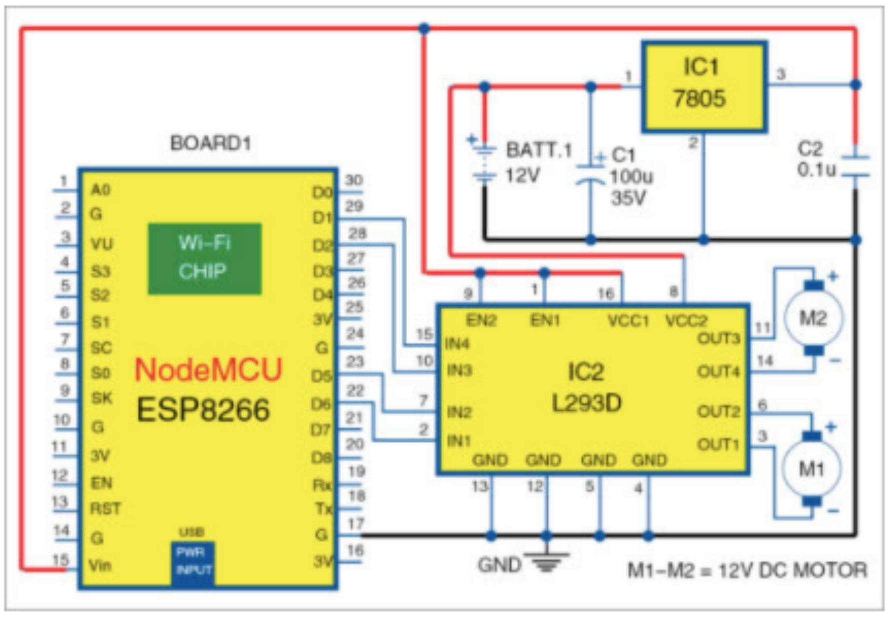


Fig. 3: Circuit diagram of the IoT robot

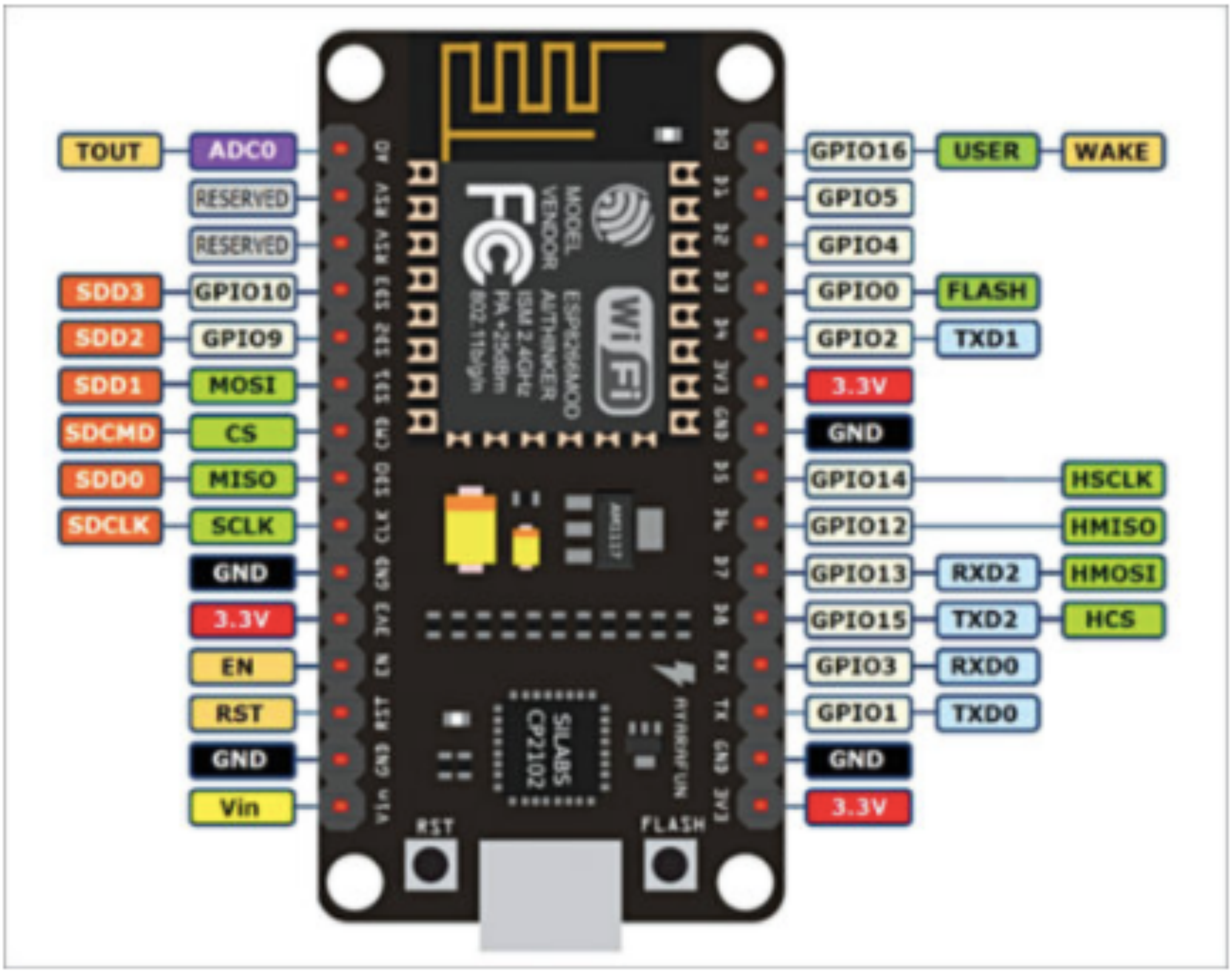


Fig. 4: NodeMCU module pin details

Blynk app

Blynk app is used for controlling the robot from anywhere in the world. It is designed for IoT applications for smart controllers using Arduino, NodeMCU and Raspberry Pi. Various sensors including temperature, humidity, pressure and accelerometer are supported by the app.

Blynk app is simple to use, flexible and has input/output features like gauge, slide bar, joystick, etc. It has less programming but requires specific Blynk header files for interfacing. Header files and libraries are easily available on GitHub.

By adding the library in Arduino IDE, you can compile the code and program NodeMCU easily.

The joystick in Blynk app is set between 0 and 255, which are the digital reference values. When the joystick moves up and down, digital values vary between 0 and 255.

When it moves along horizontal axis, voltage at x pin changes. Similarly, when it moves along vertical axis, voltage at y pin changes.

Hence, there are four directions (-x, +x, -y and +y) of joystick outputs. When the joystick moves, voltage on each output pin goes high or low depending on direction. In neutral (central) position, the joystick shows both x and y values as 128 on Blynk app.

Blynk app development

The steps for running the setup are given below.

1. Download Blynk app from Play Store and install it on your smartphone.

2. Either login or signup. You can use your current Gmail account for this.

3. Give a name to your project, say, Low-Cost IoT Robot. Select NodeMCU board from Choose Device, and Wi-Fi in Connection Type to create a new project (Fig. 5).

4. An authentication token will be sent to your registered e-mail address. Note that down as you will need it in Arduino source code (Blynkrobo.ino) later.

5. Open Arduino IDE and add the required library from this link. Extract and paste it to libraries folder in Arduino.

6. Open Blynkrobo.ino code using Arduino IDE. Copy the authentication token sent to your e-mail and paste it in the code as shown in the following lines. Replace ssid and password with your Wi-Fi's

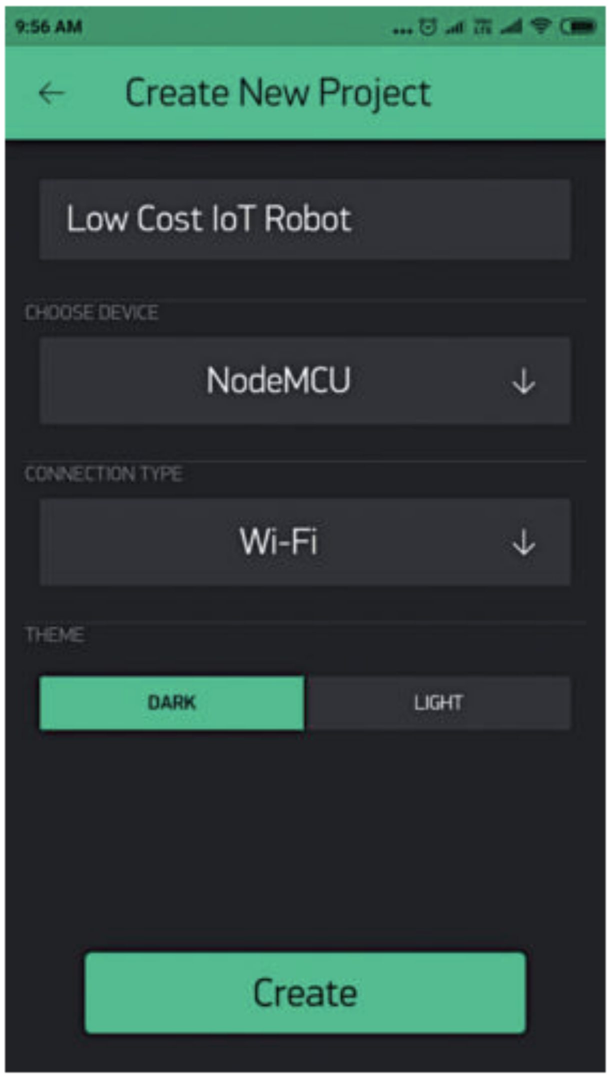

Create New Project
Low Cost IoT Robot
CHOOSE DEVICE
NodeMCU
CONNECTION TYPE
Wi-Fi
THEME

Fig. 5: Create a project

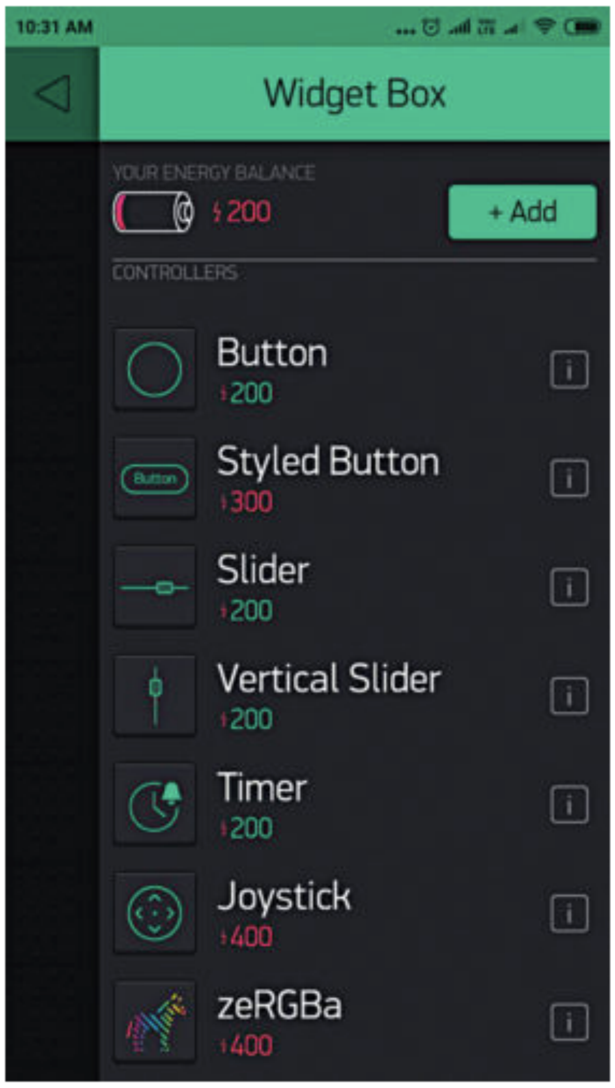

Widget Box
YOUR ENERGY BALANCE 200
CONTROLLERS
Button 200
Styled Button 300
Slider 200
Vertical Slider 200
Timer 200
Joystick 400
zeRGBa 400

Fig. 6: Widget box

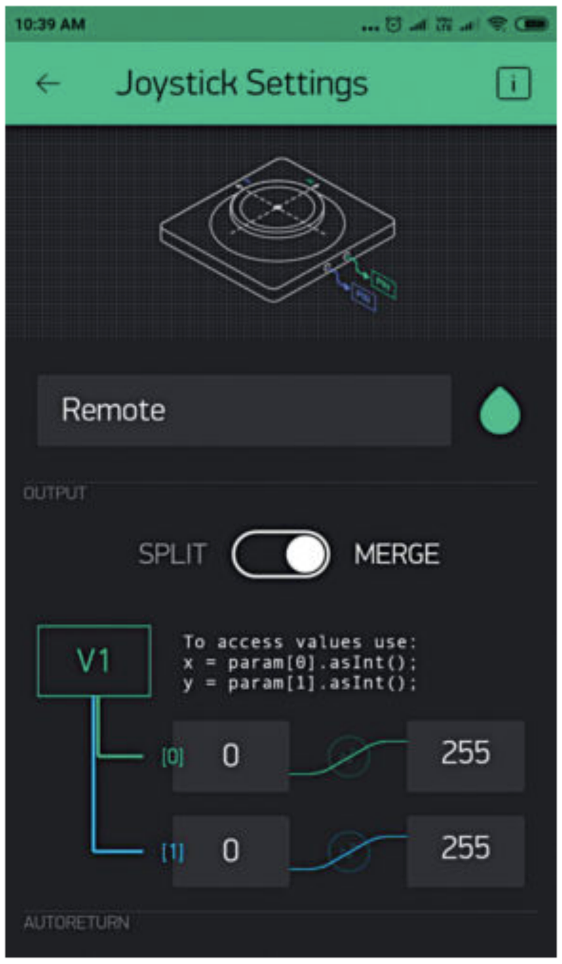
SSID and password.

char auth[]="81ee172dab924b808f73c14be0e3f2aa";

7. Connect NodeMCU with your PC/laptop using a USB cable. Under Tools, select NodeMCU 1.0 (ESP-12E Module) board and com port. Then compile and upload the source code to the board. After uploading the code, open serial monitor to check whether NodeMCU board is connected to Wi-Fi or not.

8. Go back to the created project in Blynk app. Select Joystick from Widget Box and click on Joystick to configure it (Fig. 6). Select V1 pin, and define x and y axis range from 0 to 255 (Fig. 7).

9. Press Play located at the top-right of the screen.



Remote

OUTPUT

SPLIT MERGE

V1

```
To access values use:
x = param[0].asInt();
y = param[1].asInt();
```

[0] 0 255

[1] 0 255

AUTORETURN

Fig. 7: Joystick settings

A graphical joystick panel in Blynk app will appear, as shown in Fig. 8. This panel provides digital outputs that keep changing according to the direction you move the joystick to, using your fingers. The joystick has two axes: x and y. Exact direction of the movement is interpreted through the microcontroller (NodeMCU) and the IoT (Blynk).

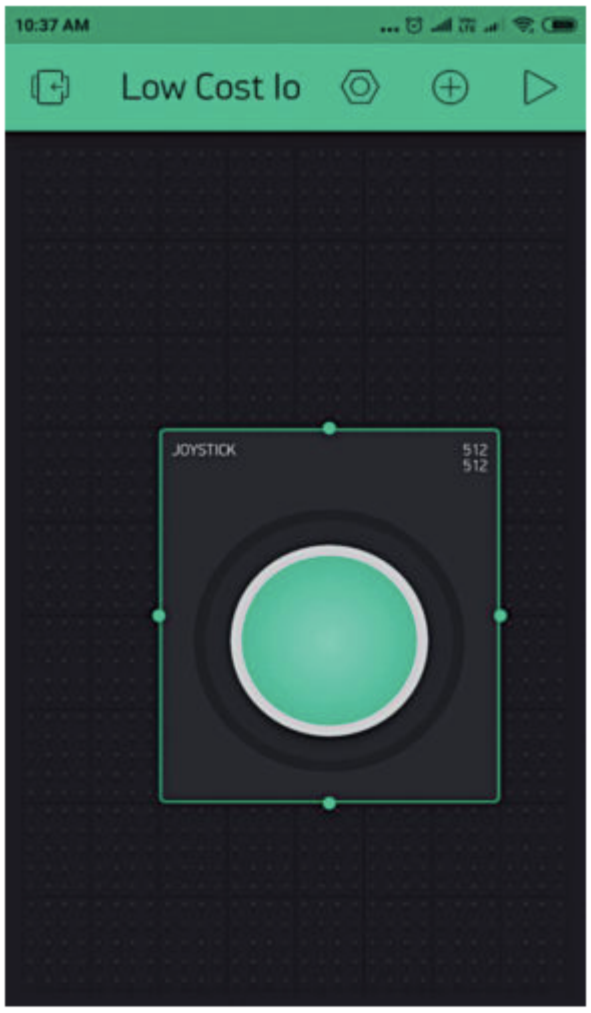


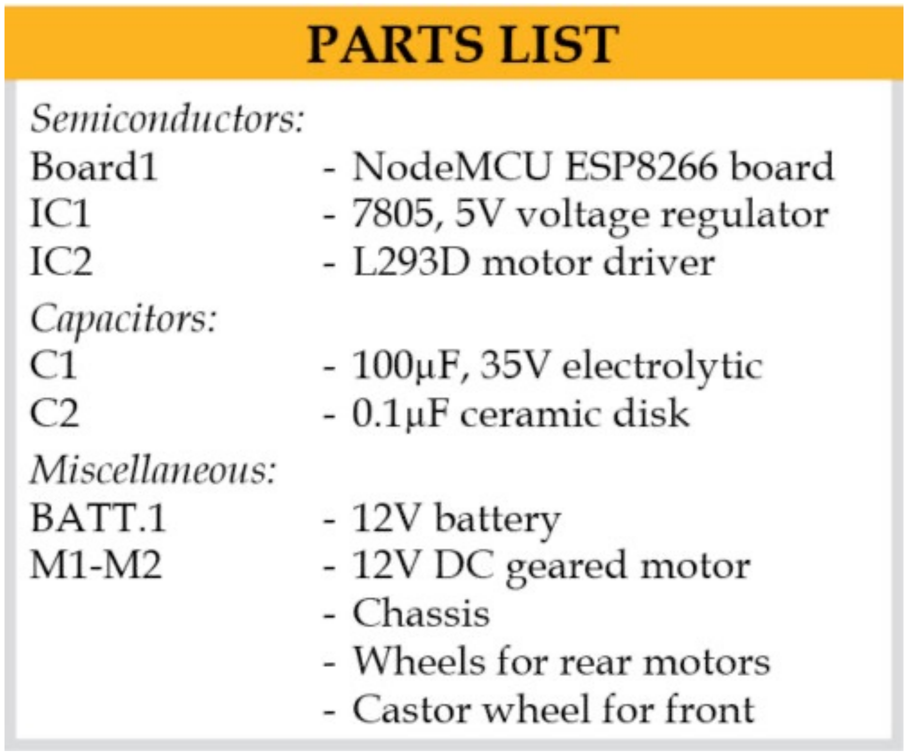
Fig. 8: Final widget joystick panel

Software

Circuit operation is done using the software program loaded into the internal memory of NodeMCU. Arduino IDE 1.6.8 is used to compile and upload the program. Software Blynkrobo.ino is written in Arduino language, which allows writing a code within a few lines. The program makes the device communicate with Blynk IoT Application Platform when connected to the Internet through an access point or Wi-Fi router.

To add ESP8266 (NodeMCU) in Arduino IDE, select File→Preferences and paste this link.

Open Board Manager from Tools→Board Menu and install ESP8266 platform. (Note: Do not forget to select your ESP8266 board from Tools→Board menu after installation.)



PARTS LIST

Semiconductors:	
Board1	- NodeMCU ESP8266 board
IC1	- 7805, 5V voltage regulator
IC2	- L293D motor driver
Capacitors:	
C1	- 100µF, 35V electrolytic
C2	- 0.1µF ceramic disk
Miscellaneous:	
BATT.1	- 12V battery
M1-M2	- 12V DC geared motor
	- Chassis
	- Wheels for rear motors
	- Castor wheel for front

In this project, external header files are needed for programming. This is a simple way to detect the joystick, and compare x and y values from Blynk app using simple programming functions.

Construction and testing

Wire the IoT robot circuit on the breadboard along with NodeMCU. Connect the left and right motors on the chassis of the robot. Fix a suitable castor wheel at front of the chassis.

In this project four digital I/O pins (D1 and D2, D5 and D6) of NodeMCU are connected to inputs of L293D H-bridge motor driver IC for controlling right and left motors of the robot.

Case 1

When you move the joystick with your finger in x-negative direction on the joystick panel, digital value is less than 30. In this condition, voltage at digital pin D6 is high, and voltages at digital pins D5, D1 and D2 are low. In this condition, the robot moves in left direction till you release the button on joystick panel.

Case 2

When you move the joystick in x-positive direction, digital value is less than 220, and voltages at pins D6, D5 and D1 are low, while at pin D2 it is high. In this condition, the robot moves in right direction till you release the button on joystick panel.

Case 3

When you move the joystick in y-negative direction, digital value is less than 30, voltages at pins D6 and D2 are high, while at pins D5 and D1 are low. In this condition, the robot moves in reverse/backward direction till you release the button on joystick panel.

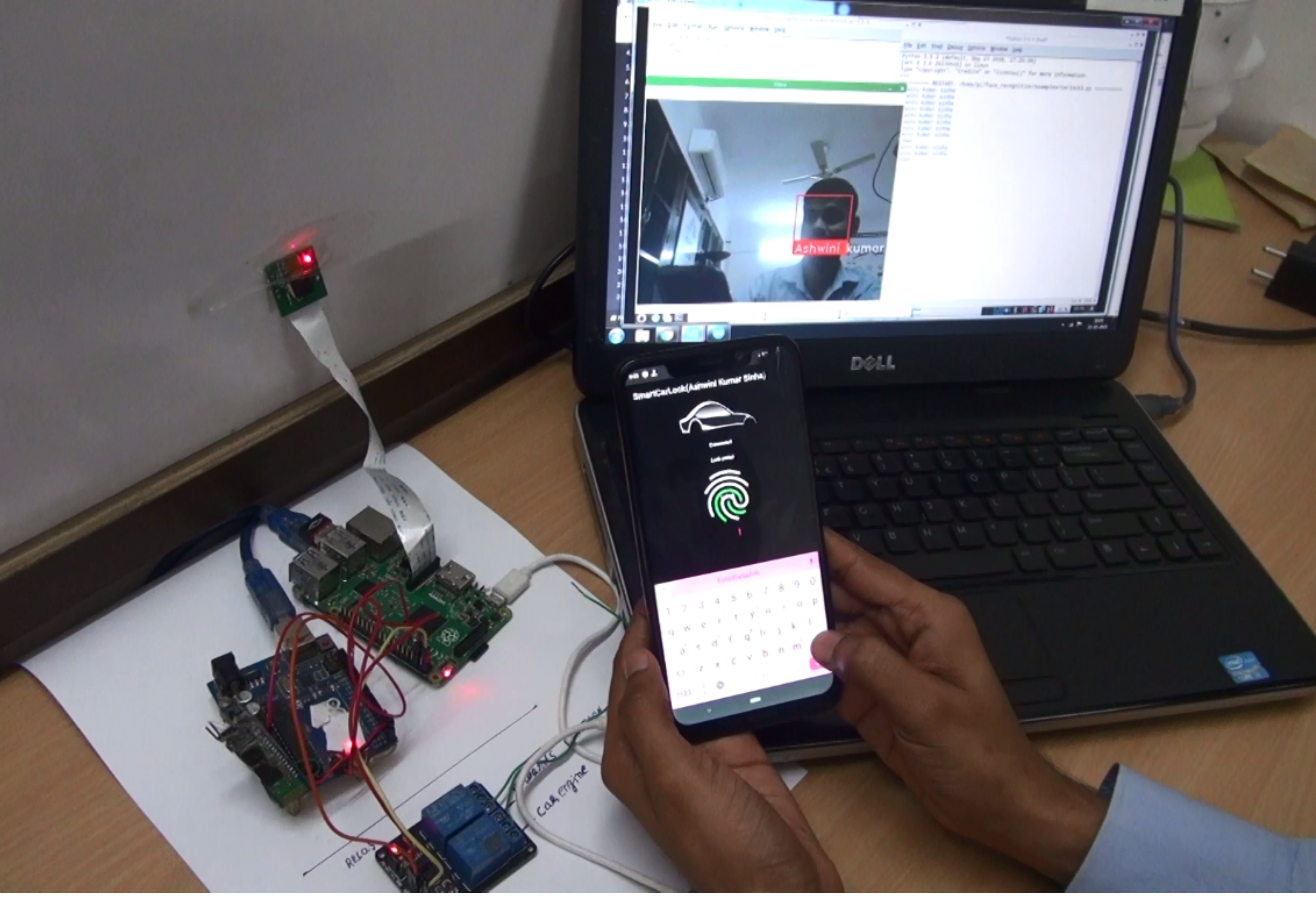
Case 4

When you move the joystick in y-positive direction, digital value is less than 220, voltages at pins D6 and D2 are low, while at pins D5 and D1 are high. In this condition, the robot moves in forward direction till you release the button on joystick panel.

Case 5

Joystick is in neutral condition. Digital values for x and y directions are 128. Here, voltages at all digital pins (D6, D5, D2 and D1) are low. In this condition, the robot is in stop condition.

For downloading source code: click here



2

3 - STEP AUTHENTICATION WITH FACE RECOGNITION, SMART LOCK FOR CAR

BY ASHWINI KUMAR SINHA

At some point, we all must have have come across news of car theft, be it in the television/newspaper or in the neighborhood. It might have occurred with some of us as well. Despite installing hi-tech security devices, thieves somehow manage to find ways to get access to your car and steal it. So, I have decided to make a prototype of a security system for automobiles which is highly efficient and fool-proof. You can also implement this security system for banks, lockers and homes where you will be protected from unauthorized access.

How it works?

1. Connect the smart lock of your car using Bluetooth.
2. Then enter the password of your car.
3. After that, a camera captures a small video of you in order to recognize your face.
4. Then it asks for your fingerprint. If correct, you'll be able to access your car, otherwise it will turn off the ignition of the car and send an alert message to its owner.

What we are going to build?

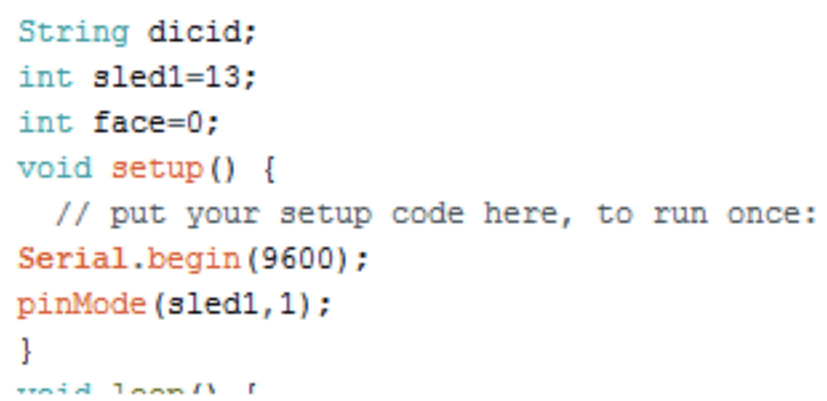
For making a 3-step car authentication smart lock, we will build a face recognition system and a biometric password so that only the authorized person gets access to the car.

Bill of materials

Component Name	Quantity	Description	Cost Approx. In INR
Raspberry Pi 4(2 GB/ 4 GB)	1	For Face Recognition	4000
RPi Camera	1	Camera	300
Arduino Uno	1	For Programming	300
Bluetooth Hc 05	1	For Wireless Connection	300
Wires	30cm	For Connection	30
Total Cost			**4,930**

Coding

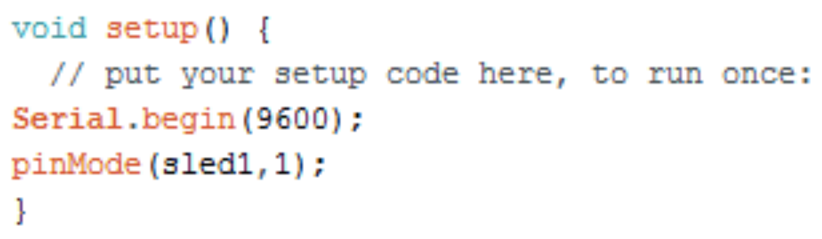
We will write an Arduino code that will enable communication between our Android app and Raspberry Pi and will thus check the lock authentication. To do so, first create variables that store the password, face detected value (1 for true and 0 for false) and pin number of the relay that controls ignition of our car. Next, we create a setup function for setting the baud rate for Bluetooth communication. Here we have used the baud rate = 9600 (default). (Refer Fig 1, Fig 2).



```
String dicid;
int sled1=13;
int face=0;
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
pinMode(sled1,1);
}
```

Fig 1. Arduino code

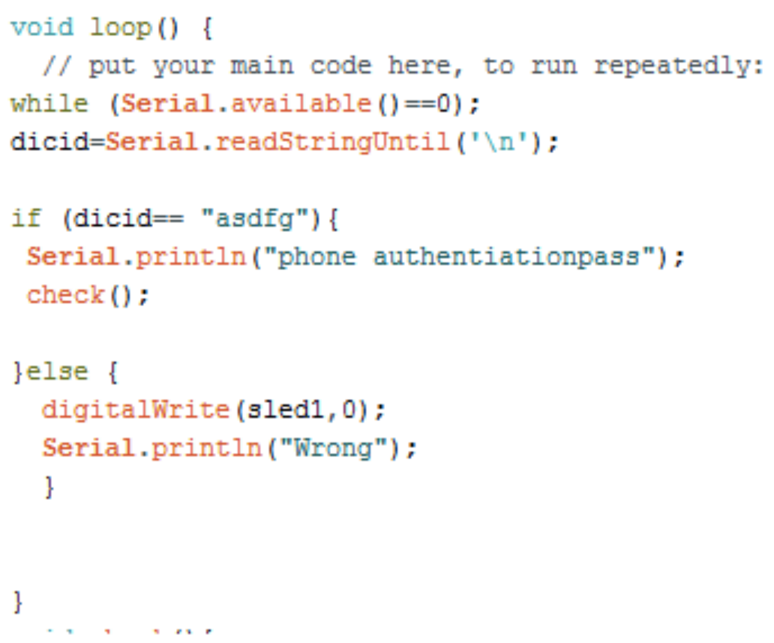
Now, we create a loop function for cross-checking the password that has been entered into the app with the one written in our code. If both the passwords match, then it will call another function called "check". The check function will basically ask Raspberry Pi for the face recognition output. (Refer Fig 3, 4)



```
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
pinMode(sled1,1);
}
```

Fig 2. Arduino code setup function.

Now we have done with our Arduino coding. Next, open the Raspberry Pi desktop window and write a Python script for face recognition.

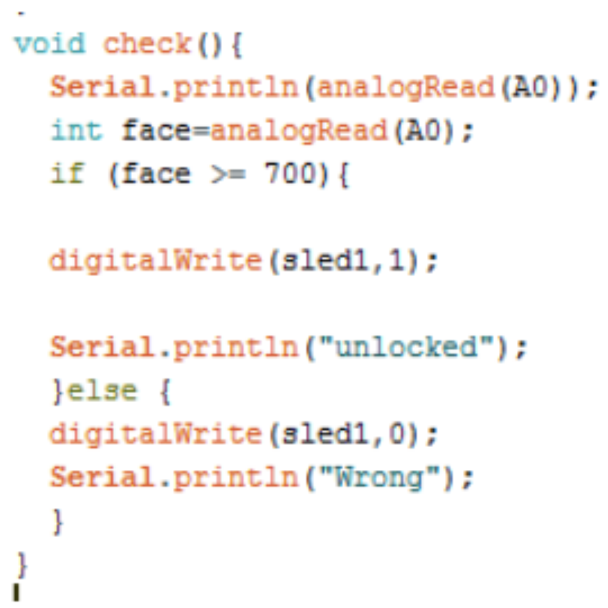


```
void loop() {
  // put your main code here, to run repeatedly:
while (Serial.available()==0);
dicid=Serial.readStringUntil('\n');

if (dicid== "asdfg"){
 Serial.println("phone authentiationpass");
 check();

}else {
  digitalWrite(sled1,0);
  Serial.println("Wrong");
  }


}
```

Fig 3. Loop function

For Face recognition:

Install dlib, pillow, cv2 and face_recognition Python modules into your Raspberry Pi. Follow the process of installation as per the instructions given in PyPl library index.

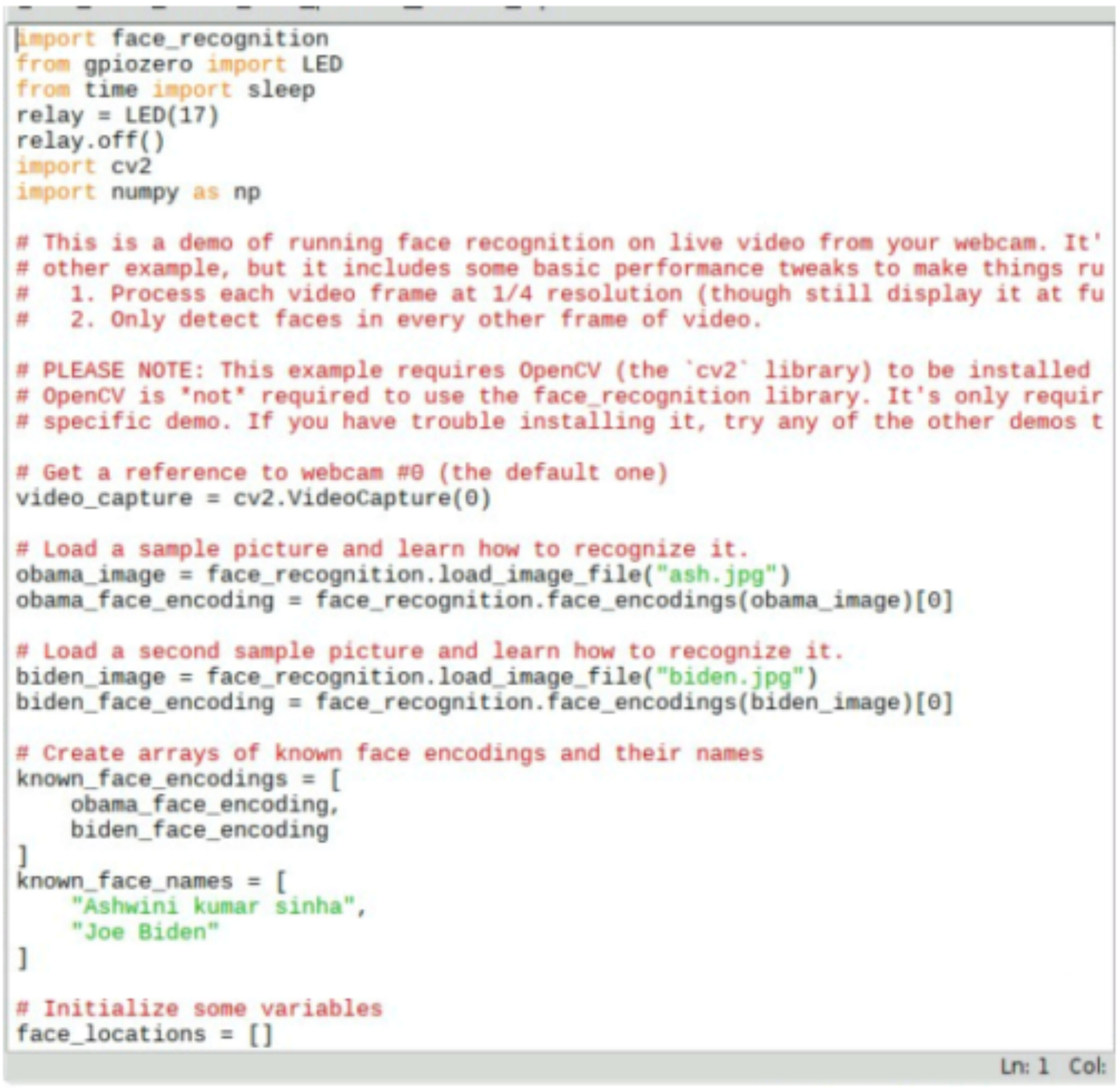


```
void check(){
  Serial.println(analogRead(A0));
  int face=analogRead(A0);
  if (face >= 700){

  digitalWrite(sled1,1);

  Serial.println("unlocked");
  }else {
  digitalWrite(sled1,0);
  Serial.println("Wrong");
  }
}
```

Fig 4. Check function

After completing the installation process, open the Python IDE and begin writing the Python script for face recognition. Make sure to include all the required modules such as face_recognition, gpiozero, cv2, numpy.

After that, paste the images of all the people who have been approved to drive your car (where you have written the script). Make sure to write the image names into the code. (Refer Fig 5.)



```
import face_recognition
from gpiozero import LED
from time import sleep
relay = LED(17)
relay.off()
import cv2
import numpy as np

# This is a demo of running face recognition on live video from your webcam. It'
# other example, but it includes some basic performance tweaks to make things ru
#   1. Process each video frame at 1/4 resolution (though still display it at fu
#   2. Only detect faces in every other frame of video.

# PLEASE NOTE: This example requires OpenCV (the `cv2` library) to be installed
# OpenCV is *not* required to use the face_recognition library. It's only requir
# specific demo. If you have trouble installing it, try any of the other demos t

# Get a reference to webcam #0 (the default one)
video_capture = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.
obama_image = face_recognition.load_image_file("ash.jpg")
obama_face_encoding = face_recognition.face_encodings(obama_image)[0]

# Load a second sample picture and learn how to recognize it.
biden_image = face_recognition.load_image_file("biden.jpg")
biden_face_encoding = face_recognition.face_encodings(biden_image)[0]

# Create arrays of known face encodings and their names
known_face_encodings = [
    obama_face_encoding,
    biden_face_encoding
]
known_face_names = [
    "Ashwini kumar sinha",
    "Joe Biden"
]

# Initialize some variables
face_locations = []
```

Fig 5. Face recognition

Now download the code and run on Raspberry Pi for face detection.

Making the app

Now open Kodular (go to www.kodular.io) and sign in. To make the UI of the app, add below given virtual components. (Refer Fig 6.)

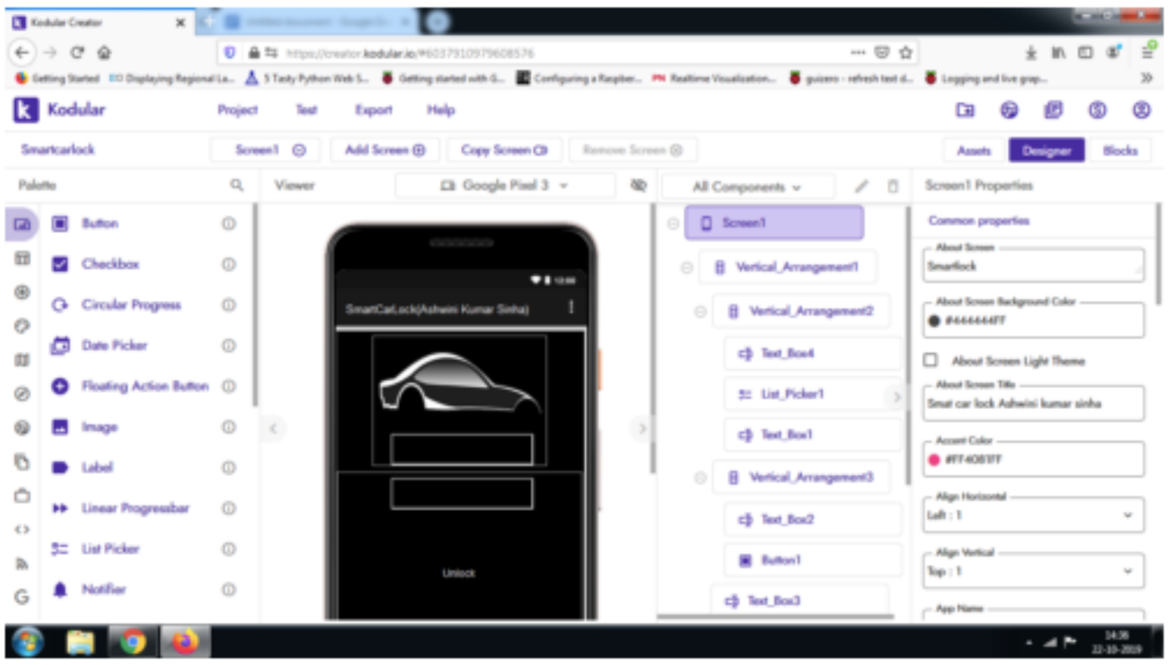
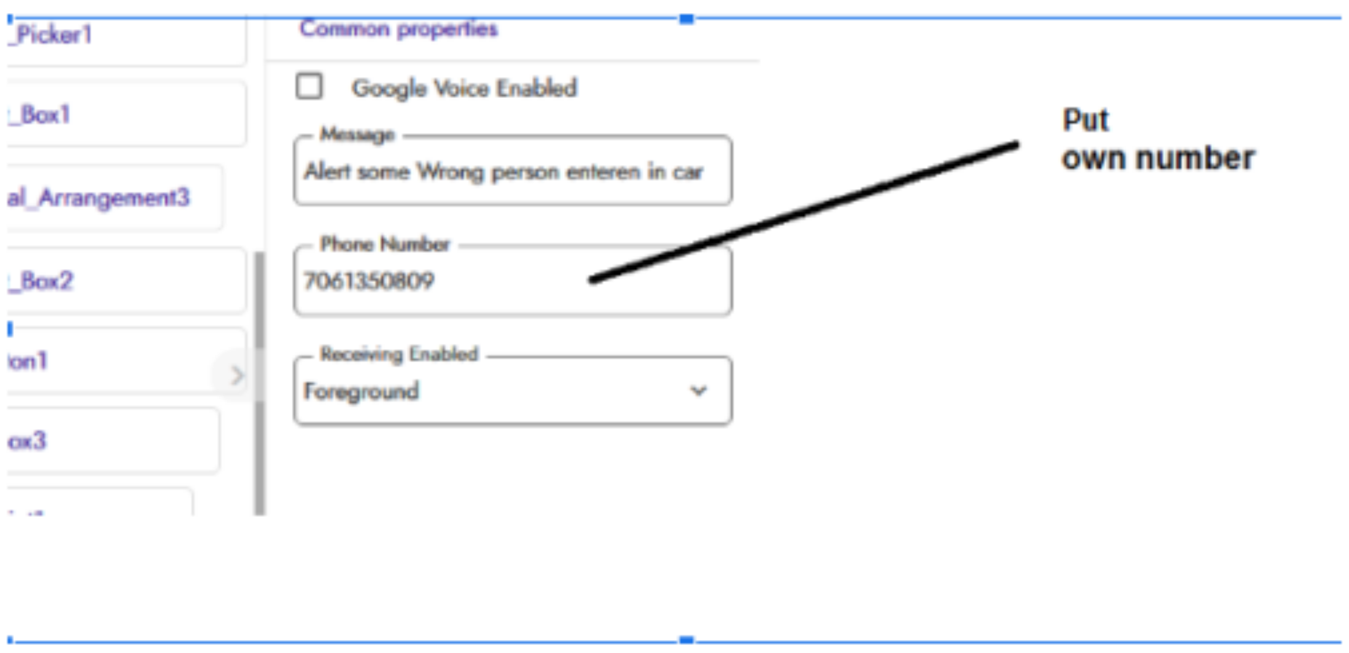


Fig 6.UI creation

- 4 Text_Boxes
- 1 Bluetooth_Client (select Connectivity -> Bluetooth Client)

- 1 List_Picker (select User Interface->List Picker)
- 1 Texting (select Social ->Texting)
- 1 Clock (select Sensors->Clock)

Now click on Texting and enter the phone number with the desired text message. (Refer Fig 7.)



Google Voice Enabled ☐

Message: Alert some Wrong person enteren in car

Phone Number: 7061350809

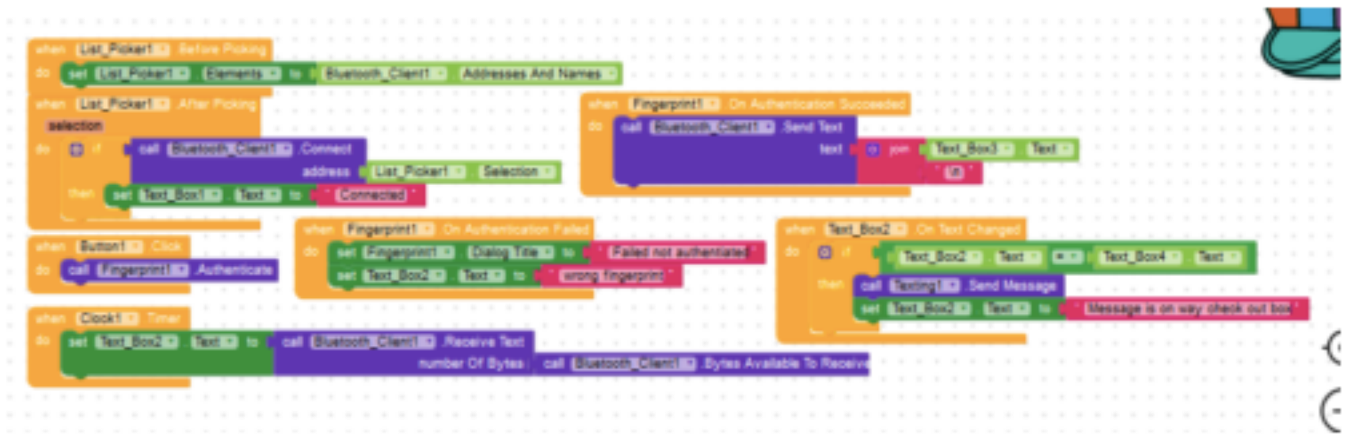
Receiving Enabled: Foreground

Fig 7. Texting settings.

Programming the app

For programming, go to Blocks and join the code blocks as shown. (Refer Fig 8.)

And finally, export the APK to your computer and then install it into your Android phone.



```
when List_Picker1 .Before Picking
do  set List_Picker1 . Elements to  Bluetooth_Client1 . Addresses And Names

when List_Picker1 .After Picking
 selection
do  if  call Bluetooth_Client1 .Connect
             address  List_Picker1 . Selection
    then  set Text_Box1 . Text to " Connected "

when Button1 .Click
do  call Fingerprint1 .Authenticate

when Clock1 .Timer
do  set Text_Box2 . Text to  call Bluetooth_Client1 .Receive Text
                                 number Of Bytes  call Bluetooth_Client1 .Bytes Available To Receive

when Fingerprint1 .On Authentication Succeeded
do  call Bluetooth_Client1 .Send Text
                      text  join  Text_Box3 . Text
                                  " lit "

when Fingerprint1 .On Authentication Failed
do  set Fingerprint1 . Dialog Title to " Failed not authentiated "
    set Text_Box2 . Text to " wrong fingerprint "

when Text_Box2 .On Text Changed
do  if  Text_Box2 . Text = Text_Box4 . Text
    then  call Texting1 .Send Message
          set Text_Box2 . Text to " Message is on way check out box "
```

Fig 8. Code blocks.

Download **Source Folder**

Connection

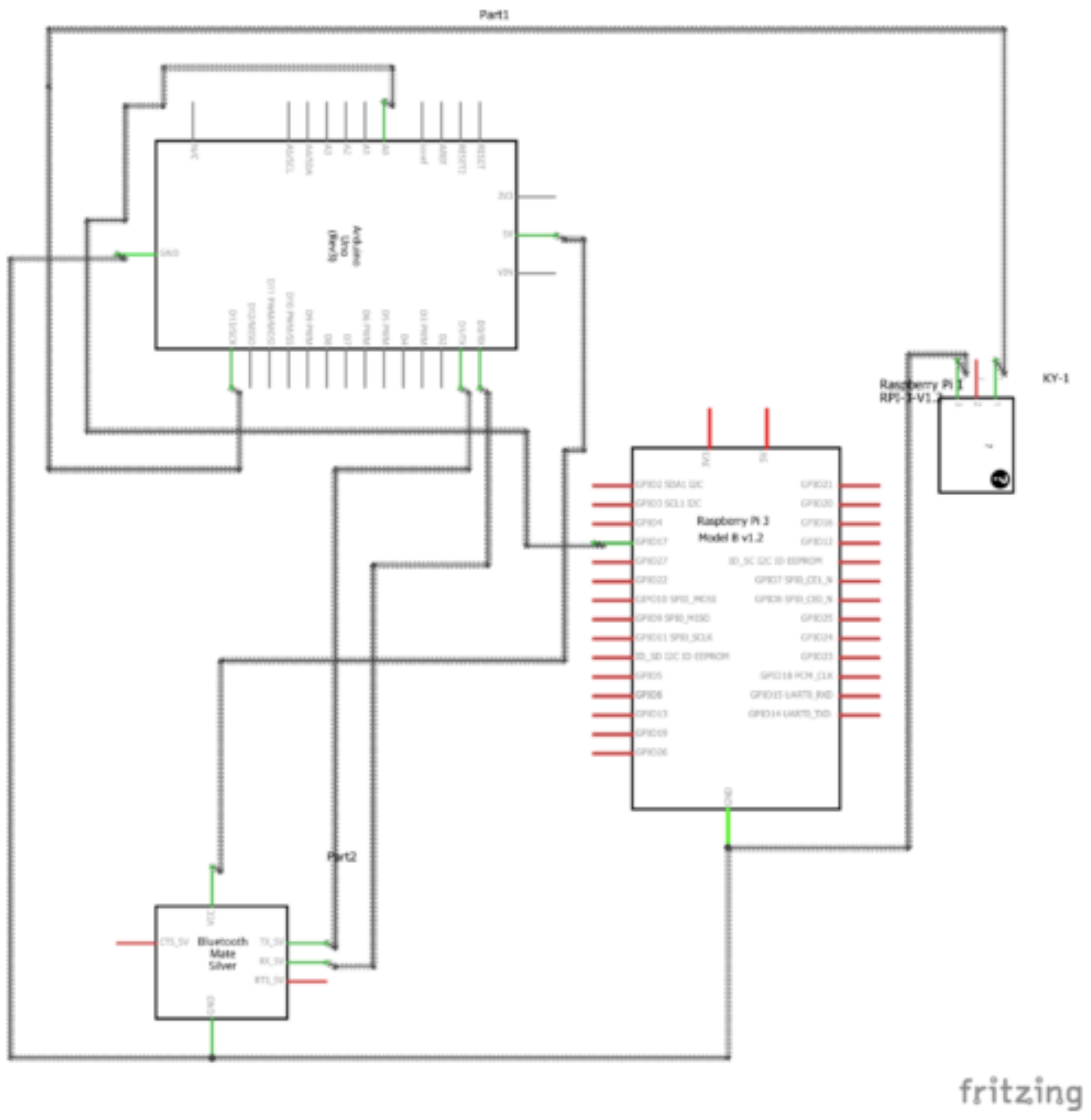


Fig 10. Circuit Diagram.

Connect the components as follows:

Arduino RX------------------HC 05 TX

Arduino TX------------------HC 05 RX

Arduino 5v ------------------HC 05 VCC

Arduino GND---------------GND

Arduino A0-------------------Raspberry GPIO Face pin Out

Arduino Pin 13--------------Relay

Testing the app

To test our prototype, open the app and turn on the Bluetooth of your phone. If app asks for access to text messages, then allow that. Now power the Raspberry Pi and run the carface.py script. When this script runs, the camera detects your face and opens it in a new window. If the camera detects the correct face, it will then send a verification signal to Arduino.

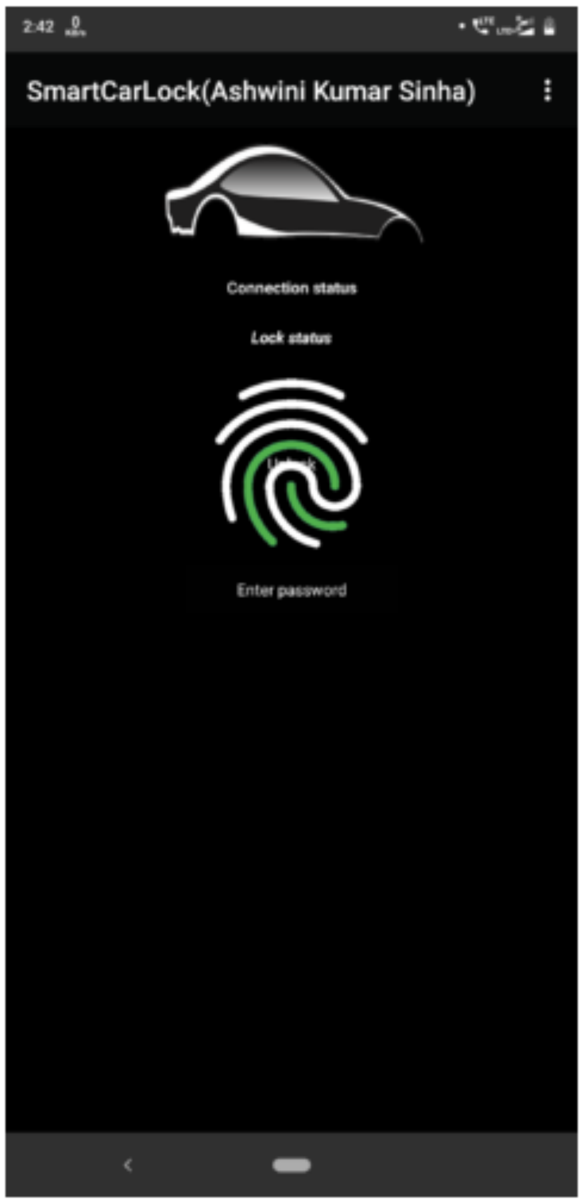


Fig.9. Final App

Next, connect the car with the app by tapping on the car icon.

Now, select Bluetooth HC 05 and after successful connection, enter the password and your fingerprint by tapping the fingerprint icon. If Arduino detects all process to be correct, then it triggers the relay which then sets the car ignition to ON. If incorrect password/ fingerprint/ face is detected, then the car ignition is set OFF and the app sends an alert message to the owner to inform about the unauthorised access.

You can watch Video for this DIY here: https://youtu.be/ozbKsDRSpwU

3

HOME AUTOMATION SYSTEM USING A SIMPLE ANDROID APP

BY KARTHIK RAJASEKARAN

Nowadays, people have smartphones with them all the time. So it makes sense to use these to control home appliances. Presented here is a home automation system using a simple Android app, which you can use to control electrical appliances with clicks or voice commands. Commands are sent via Bluetooth to Arduino Uno. So you need not get up to switch on or switch off the device while watching a movie or doing some work.

Home automation: circuit and working

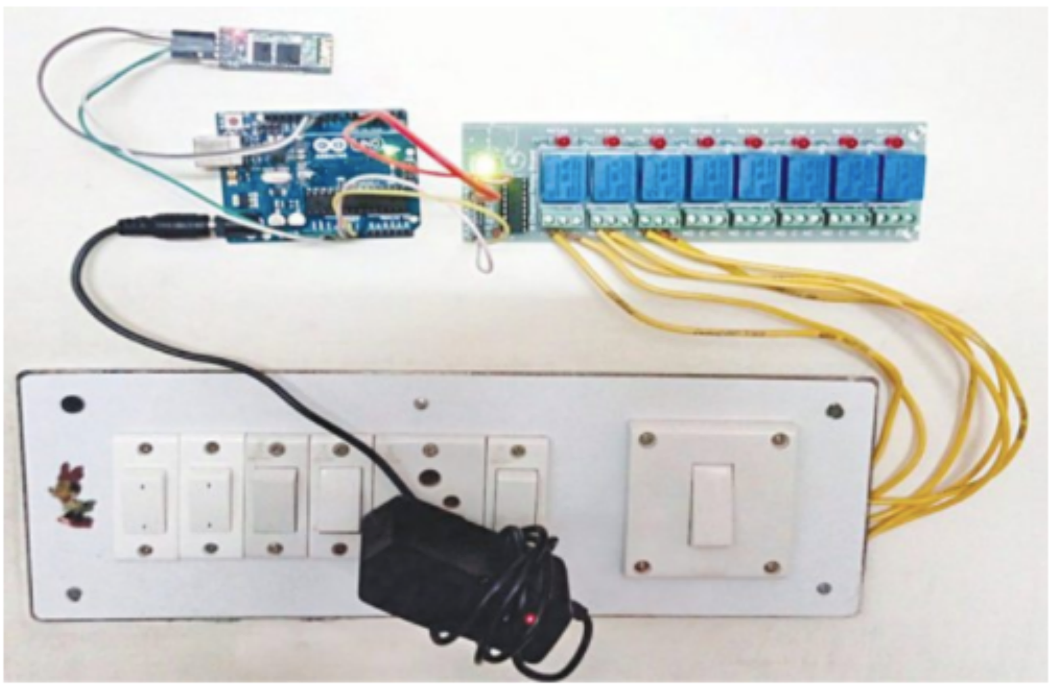


Fig. 1: Author's prototype

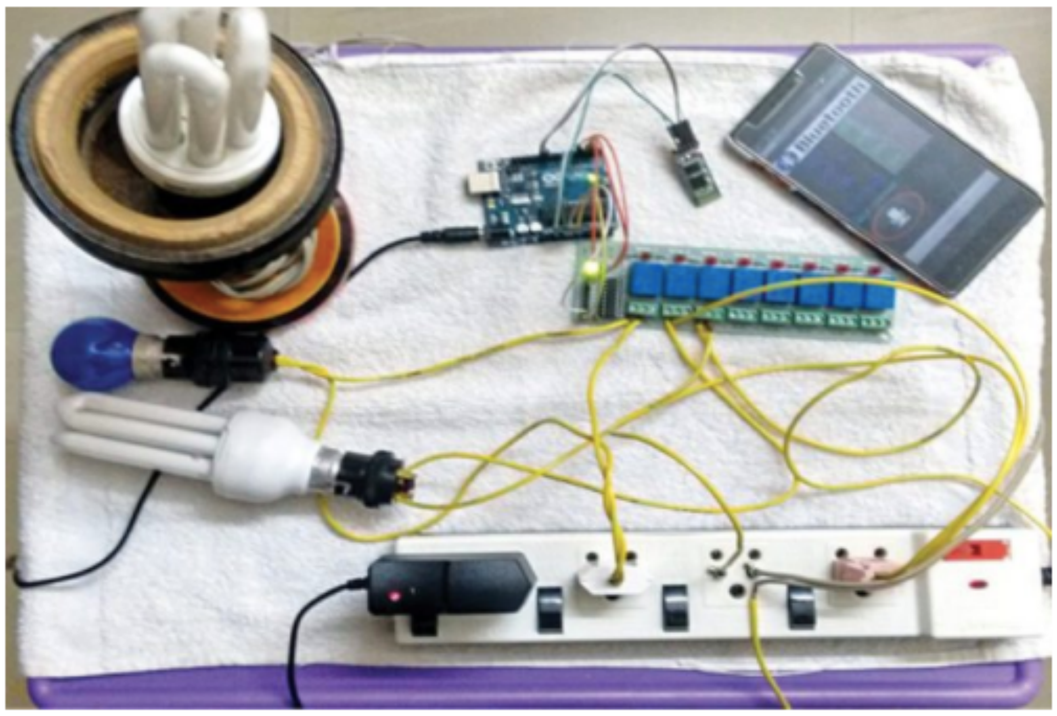


Fig. 2: Connections for the appliances

The home automation circuit is built around an Arduino Uno board, Bluetooth module HC-05 and a 3-channel relay board. The number of channels depends on the number of appliances you wish to control. Arduino Uno is powered with a 12V DC adaptor/power source. The relay module and Bluetooth module can be, in turn, powered using a board power supply of Arduino Uno. Author's prototype is shown in Fig. 1. Connection details for each appliance are shown in Fig. 2.

Bluetooth module

Bluetooth module used in this project is HC-05 (Fig. 4), which supports master and slave mode serial communication (9600-115200 bps) SPP and UART interface. Using these features it can communicate with other Bluetooth-enabled devices like mobile phones, tablets and laptops. The module runs on 3.3V to 5V power supply.

Relay module

A relay allows you to turn on or turn off a circuit using voltage and/or current much higher than what Arduino could handle. Relay provides complete isolation between the low-voltage circuit on Arduino side and the high-voltage side controlling the load. It gets activated using 5V from Arduino, which, in turn, controls electrical appliances like fans, lights and air-conditioners. An 8-channel relay module is shown in Fig. 5.

Arduino Uno board

Arduino is an open source electronics prototyping platform based on flexible, easy-to-use hardware and software. It is intended for artists, designers, hobbyists and anyone interested in creating interactive objects or environments.

Arduino Uno is based on ATmega328 microcontroller (MCU). It consists of 14 digital input/output pins, six analogue inputs, a USB connection for programming the onboard MCU, a power jack, an ICSP header and a reset button. It is operated with a 16MHz crystal oscillator and contains everything needed to support the MCU. It is very easy to use as you simply need to connect it to a computer using a USB cable, or power it with an AC-to-DC adapter or battery to get started. The MCU onboard is programmed in Arduino programming language using Arduino IDE.

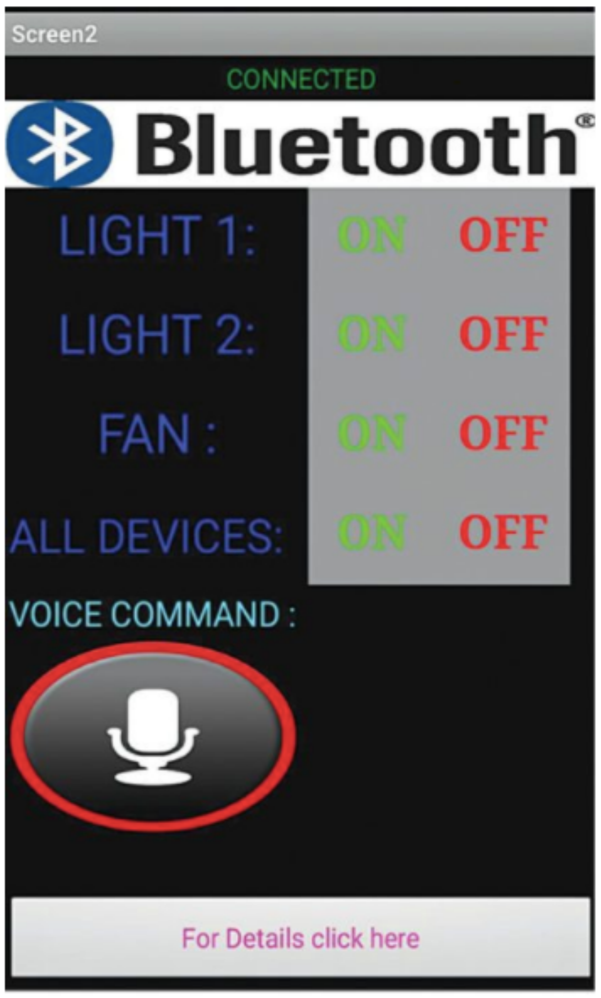


Fig. 3: Control panel on Android smartphone

Fig. 4: Bluetooth module

In this home automation project circuit, Pins 10 and 11 of Arduino are connected to pins TXD and RXD of the Bluetooth module, respectively, as shown in Fig. 6.

Pins Gnd and Vcc of the Bluetooth module are connected to Gnd and +3.3V of Arduino board respectively. Pins 2, 3 and 4 are connected to the three relays (RL1, RL2 and RL3) of the relay board. Pins Vin and Gnd of the relay board are connected to pins Vin and Gnd of Arduino board, respectively.

Note. Vin is usually used to give input power, but since we are supplying 12V to Arduino using an adaptor, we can use Vin pin on Arduino to power the 12V relay module.

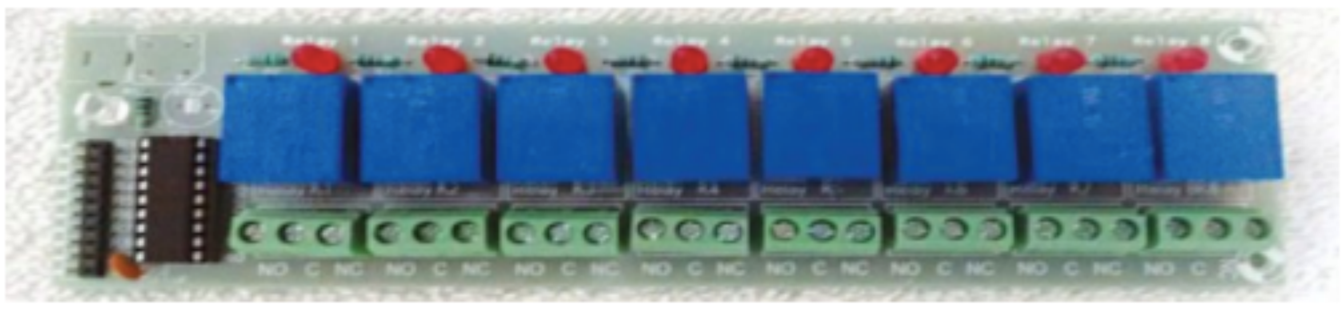


Fig. 5: An 8-channel relay module

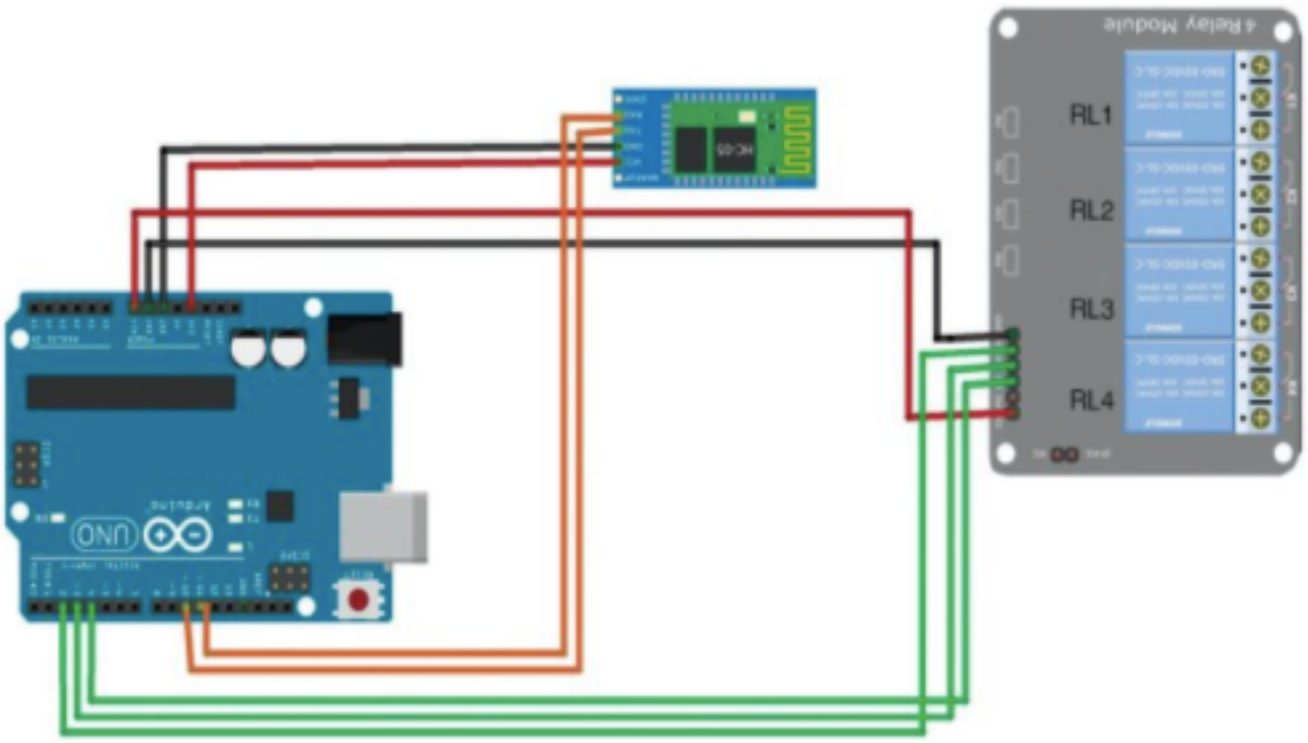


Fig. 6: Relay module connection

Software

The software program for the home automation project(homeautomation.ino) is written in Arduino programming language called Processing. Arduino Uno is programmed using Arduino IDE software that you can download from arduino.cc. MIT App Inventor software was used to create the Android app (.apk) for this project.

The app on your smartphone sends data when you click on buttons or feed voice commands via Bluetooth in the mobile to Bluetooth module HC-05 connected with Arduino board. Received data pin TXD of the HC-05 is connected to Arduino. Arduino Uno processes the received data and controls the relay board accordingly.

Procedure for installing the Android app (.apk) is as follows:
1. Download the app (homeautomation.apk).
2. Run .apk file. It will prompt you to complete the action. Click Package Installer and then Install.

3. You will also need a voice-recognition app on your Android smartphone. Most smartphones have this app preinstalled. If you do not have it, download one from Google Play Store.

Download source folder: click here

Construction and testing

Assemble the circuit as shown in the circuit diagram. Open Arduino IDE and compile the program (sketch). Upload the sketch (homeautomation.ino) to Arduino board. Switch on the power supply to Arduino by connecting it to 12V power source. Pair Bluetooth module with your Android phone. Type password '1234' (default password) of Bluetooth module.

Click Bluetooth Image on the app to connect it with the Bluetooth module. It automatically connects and displays as Connected in the app.

You are now ready to control the appliances using the app. You can either use on/off buttons or voice commands to control the appliances. You can control more electrical appliances by increasing the number of channels in the relay. For instance, using an 8-channel relay, you can control up to eight devices. For this, you need to alter the source code by adding input commands and voice commands to control the devices.

4

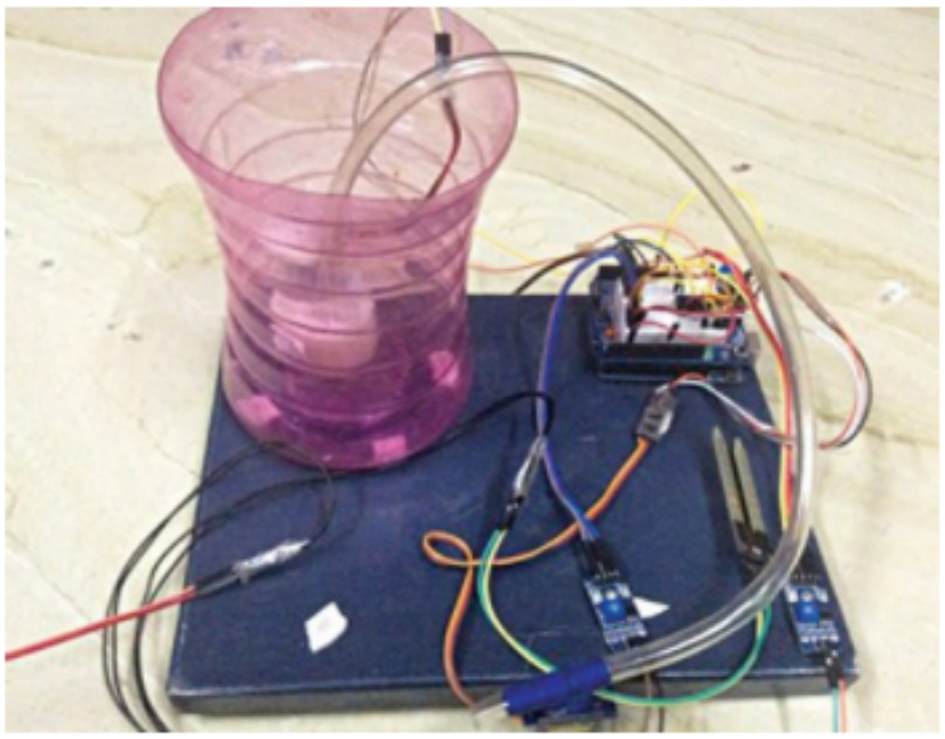
AUTOMATED PLANT WATERING SYSTEM

BY AYAN PAHWA

During summers, most people are too lazy to water the potted plants on their rooftop gardens every day. Explained in this section is a simple and exciting automatic plant watering system that you can build yourself in just a few hours. It is an Arduino based automatic plant watering system that uses a soil moisture sensor. The author's prototype is shown in Fig. 1.

Automatic plant watering system circuit and working

The circuit diagram of the automatic plant watering system is shown in Fig. 2. The circuit comprises an Arduino UNO board, a soil moisture sensor, a servo motor, a 12V water pump and an L293D (IC1) motor driver IC to run the water pump.


Author's Prototype

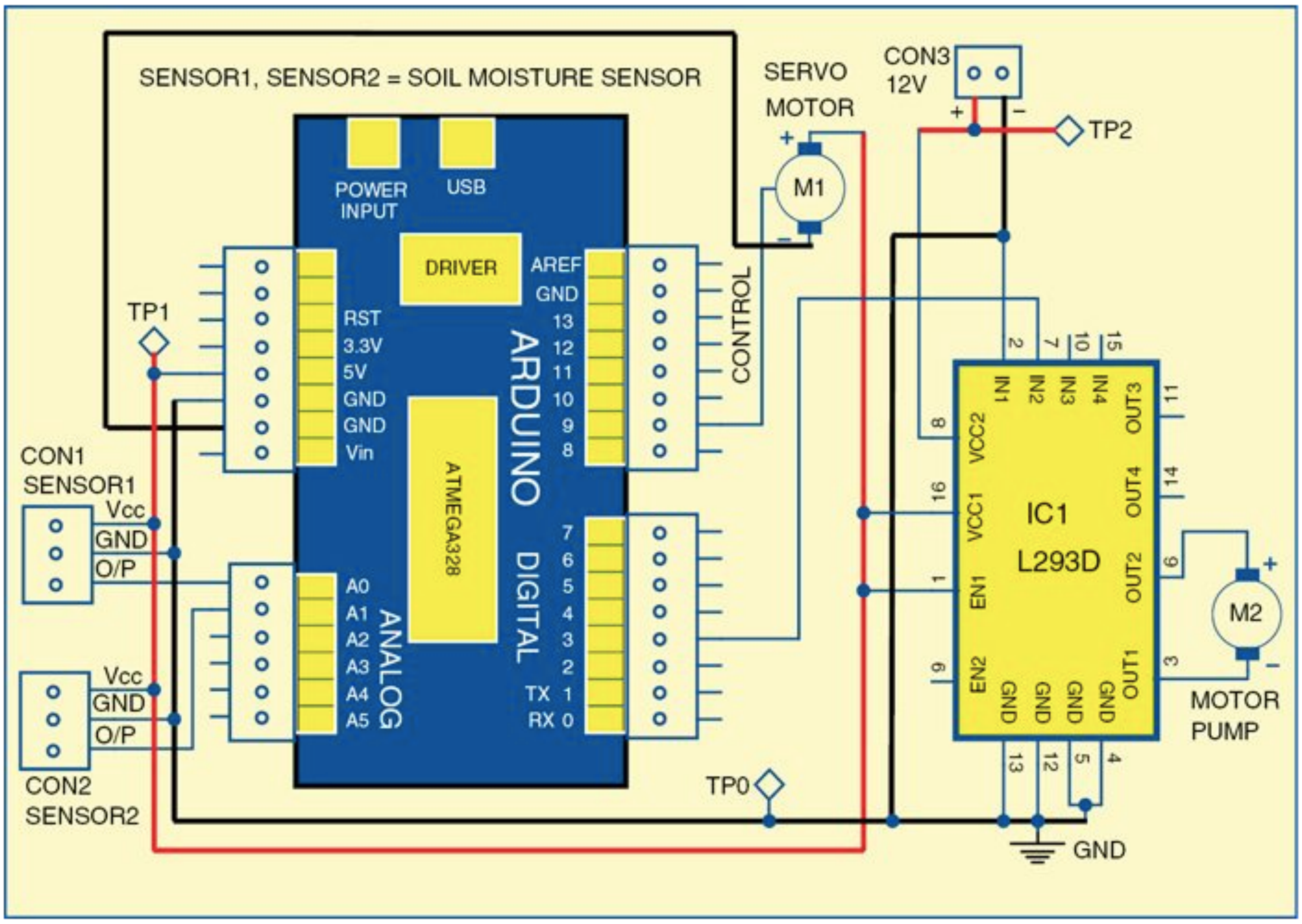


Fig. 2: Circuit diagram of the automatic plants watering system

You can power the Arduino board using a 7V to 12V wall wart or plug-in adaptor or solar panel. You need a separate 12V battery or power supply or solar panel for the pump motor.

Soil moisture sensor

Two types of soil moisture sensors are available in the market—contact and non-contact sensors. A contact soil sensor (as shown in Fig. 3) is used in this project because it has to check soil moisture to measure the electrical conductivity.

The moisture sensor provides an analogue output, which can easily be interfaced with Arduino. In this project, two sensors can be connected to analogue pins, A0 and A1, of the Arduino board. Each sensor has four pins (Vcc, Gnd, A0 and D0) available for interfacing with the Arduino board. Here, digital output pin (D0) is not used. The water pump and servo motor are controlled by Arduino connected to digital pins 3 and 9, respectively. That is, the servo motor signal control pin is connected to pin 9 of the Arduino board.

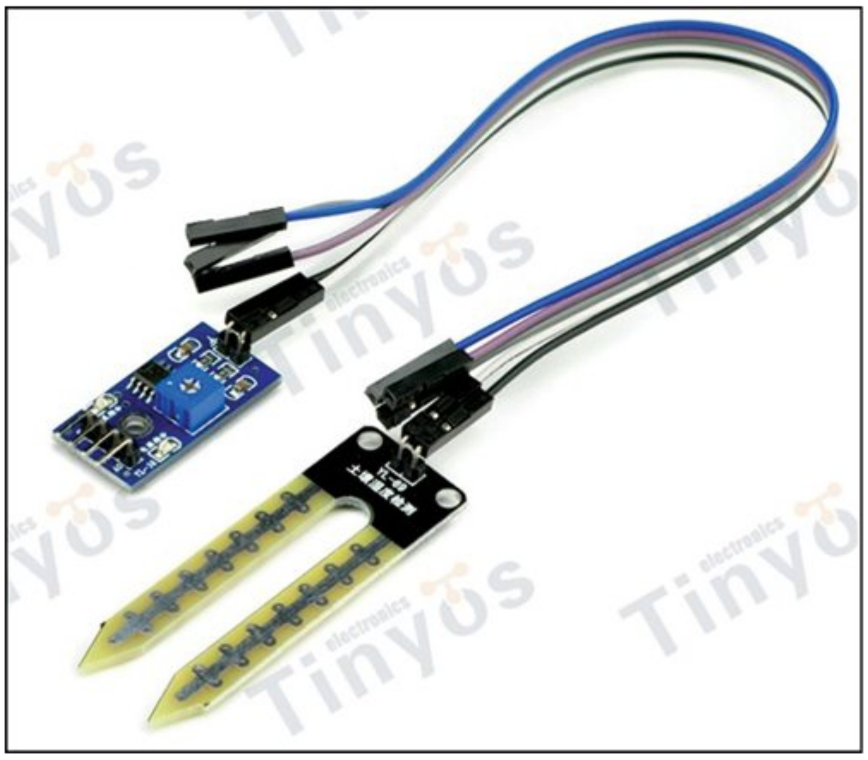


Fig. 3: Soil moisture sensor (contact type)

Fig. 4: Motor pump

The program in the Arduino reads the moisture value from the sensor every 20 seconds. If the value reaches the threshold value, the program does the following three things:

1. It moves the servo motor horn, along with the water pipe fixed

on it, toward potted plant, whose moisture level is less than the predetermined/ threshold level.

2.It starts the motor pump to supply water to the plant for a fixed period of time and then stops the water pump (refer Fig. 4).

3. It brings back the servo motor horn to its initial position.

Software program

The program is written in Arduino programming language. The code is well commented and is easy to understand. Compile the autowatering.ino code and upload it to the microcontroller, using Arduino IDE version 1.

The sensor will calibrate by itself once it is kept in the soil and the threshold value will be shown on the serial monitor in Arduino. Serial debugging is available in this program. Comment out if you do not wish to use the serial monitor.

Construction and testing

An actual-size, single-sided PCB layout of the automatic plant watering system is shown in Fig. 5 and its component layout in Fig. 6.

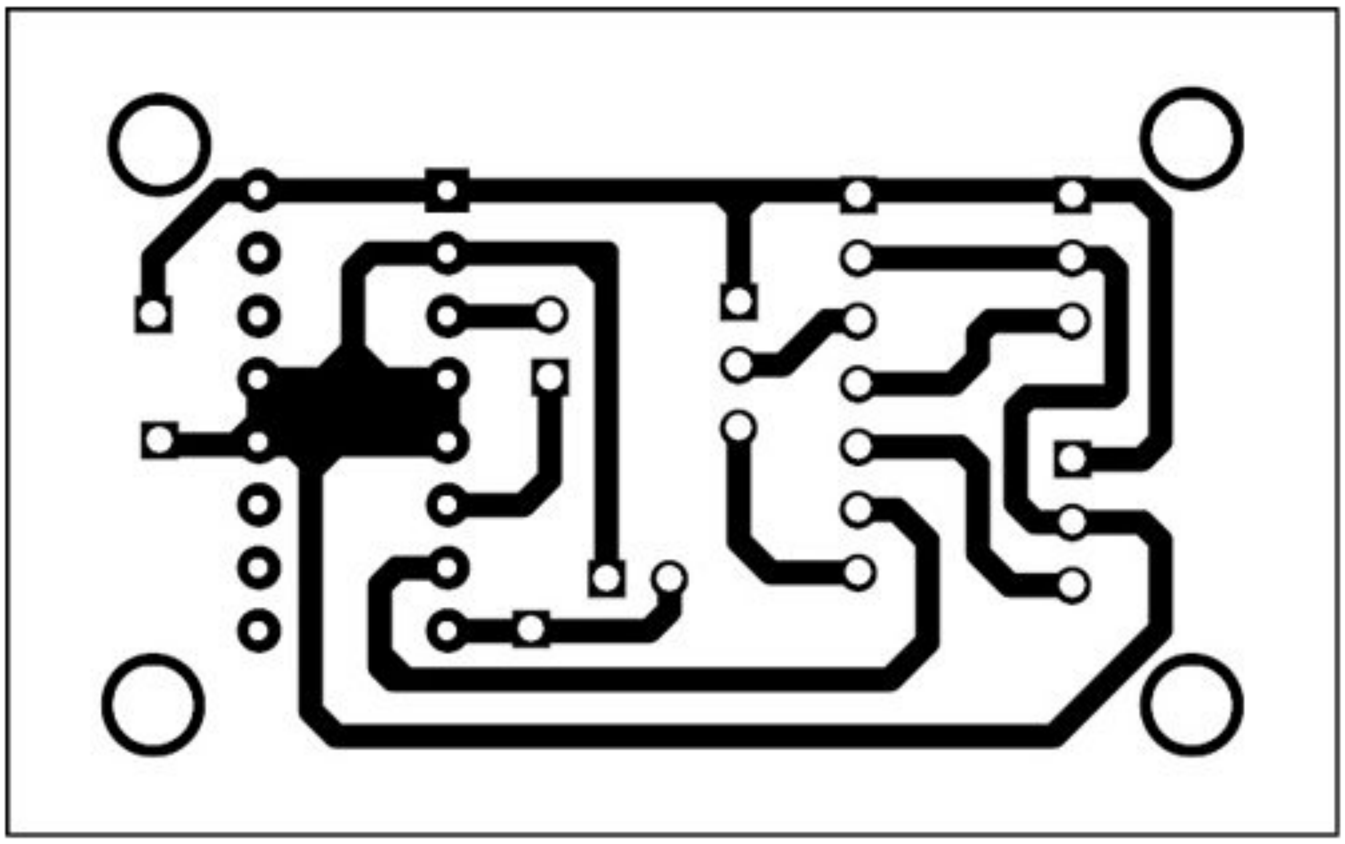


Fig. 5: Actual-size PCB layout of the circuit

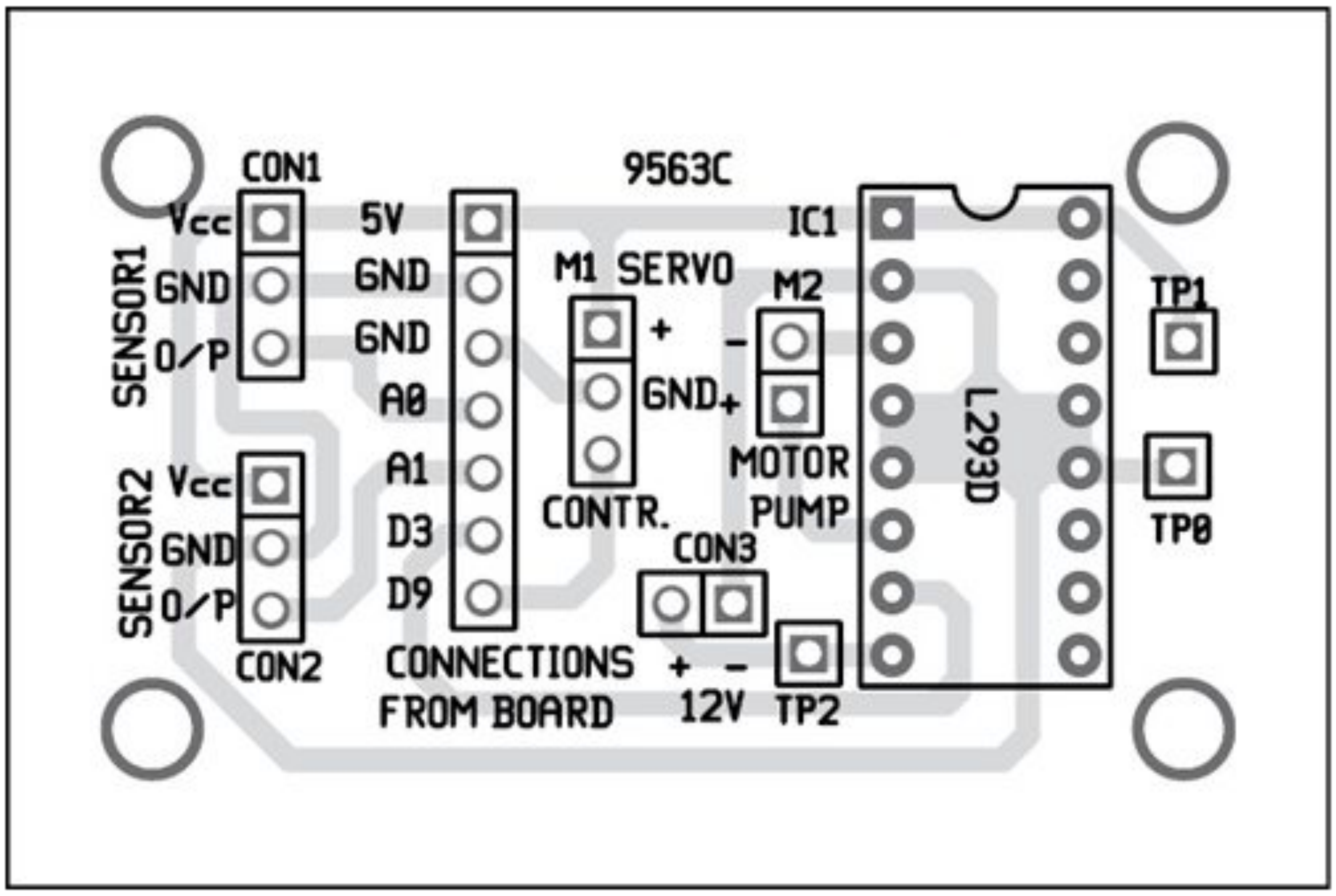


Fig. 6: Component layout of the PCB

Assemble the components on the PCB to minimise errors. Alternatively, you can assemble them on a breadboard or Arduino prototyping shield or a general-purpose PCB. Upload the code to Arduino UNO board and install the sensors in the soil of the potted plants. Do not immerse the sensors fully inside the soil.

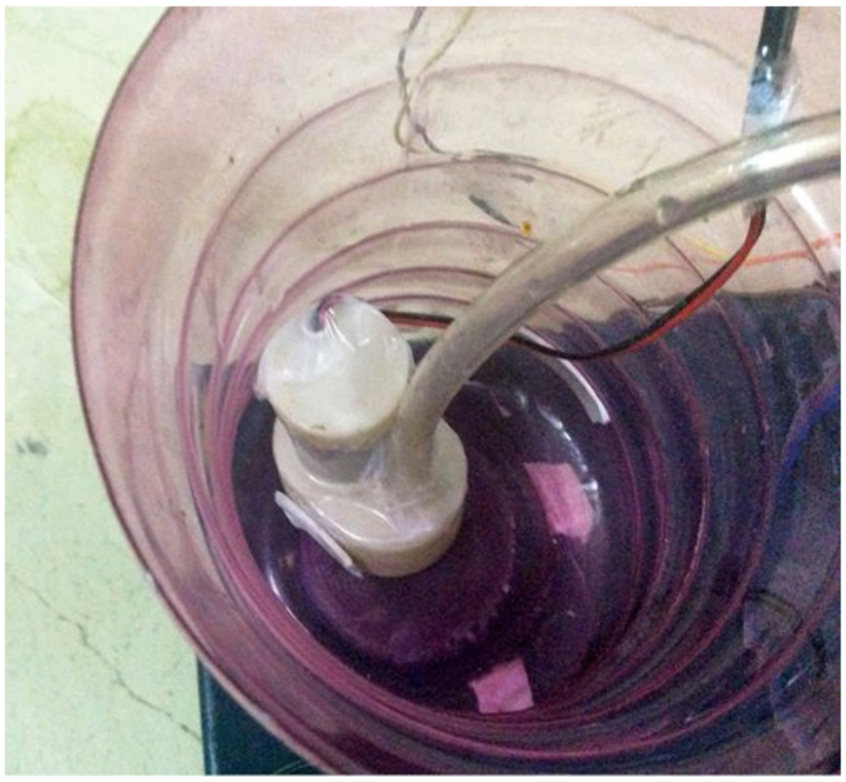


Fig. 7: Installing water pump in the container

Install the pump in a water container (refer Fig. 7) that can hold a few litres of water. attach the water pipe on the servo motor horn as shown in Fig. 8.

Before powering the circuit on, you need to keep in mind the following macro definitions in the code:

1. Changing the angle of rotation of the servo horn toward the first pot and second pot. The default values are 70 degrees and 145 degrees.

2. Changing the watering time according to the size of the pot. The default values are five seconds and eight seconds.

3. Changing the threshold value according to your need. The default value is 600.

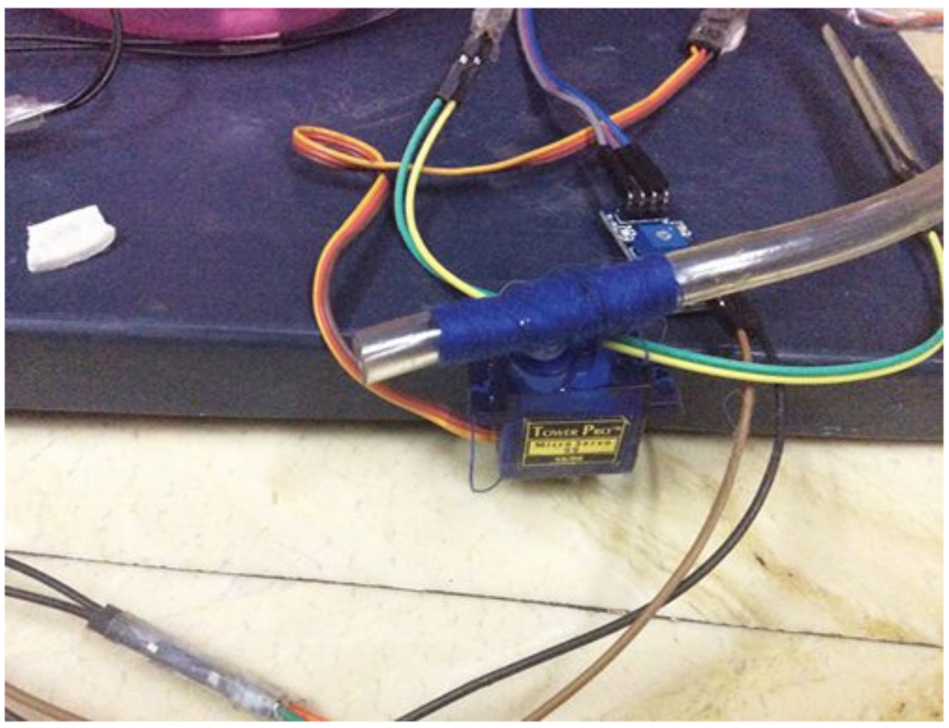
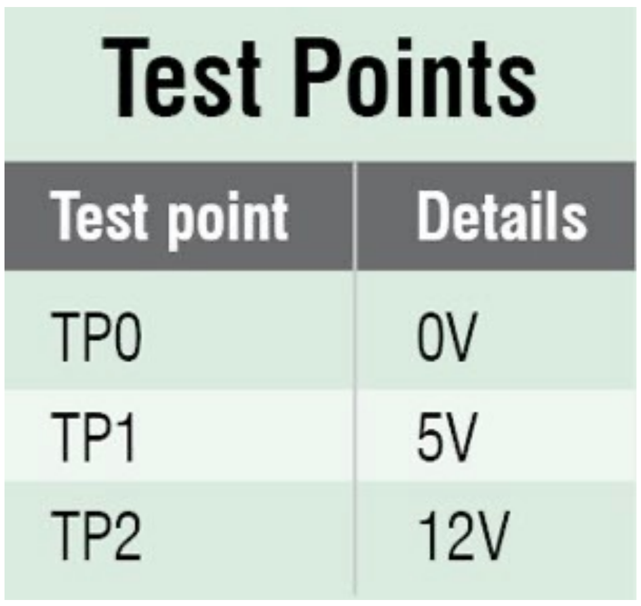


Fig. 8: Attaching the pipe on the servo horn

Place the flower pots where the pipe from the servo motor horn can easily reach them. When the moisture level dips below 600, servo horn rotates at an angle of 70 degrees. That is, after servo motor horn moves 70 degrees toward the first pot, the motor pump will be on for five seconds and then stop automatically. Then, the servo returns to its original position. Similarly, if you are using a second sensor, the servo motor horn will move to 145 degrees to the second biggest pot, motor pump will be on for eight seconds and then stop automatically. The servo returns to its original position.



Test Points

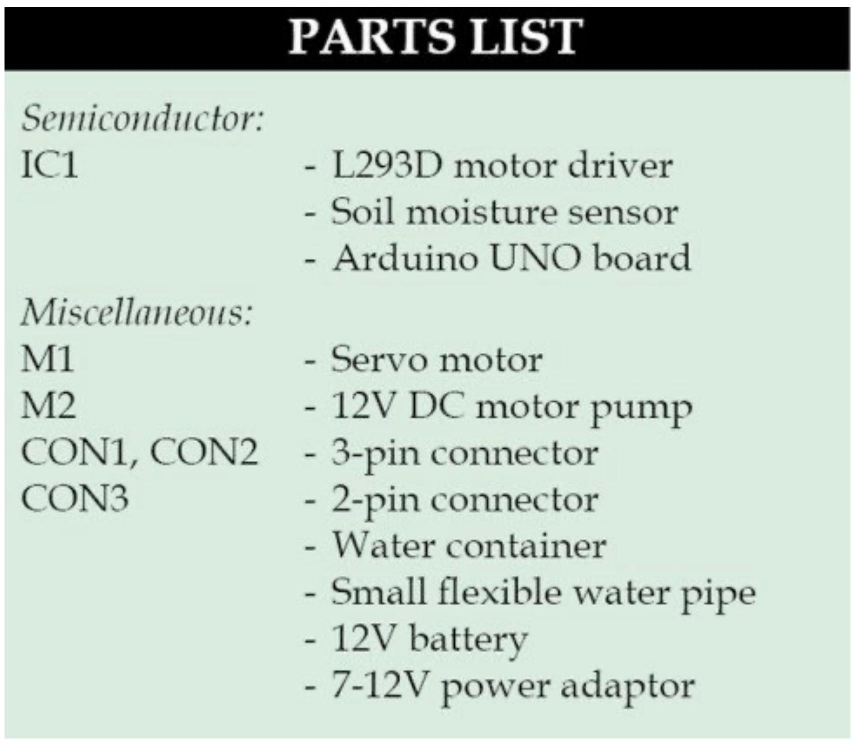
Test point	Details
TP0	0V
TP1	5V
TP2	12V

Download PCB and Component Layout PDF: Click Here

Download Source Code: Click Here

Further application

Using the Arduino UNO board, you can water six different potted plants. By adding a few more lines in the code, you can water even more plants—by using the Arduino Mega 2560 board which has more analogue input pins.



PARTS LIST

Semiconductor:	
IC1	- L293D motor driver
	- Soil moisture sensor
	- Arduino UNO board
Miscellaneous:	
M1	- Servo motor
M2	- 12V DC motor pump
CON1, CON2	- 3-pin connector
CON3	- 2-pin connector
	- Water container
	- Small flexible water pipe
	- 12V battery
	- 7-12V power adaptor

You can also add an Ethernet or Wi-Fi shield and use the Twitter library, which will tweet from your plants side to send messages like: I need water, the tank is empty, refill the tank, thanks for the water, and so on.

A 16×2 LCD can be added to indicate moisture levels.

You can also enable the circuit to refill the tank after a few days, depending on the volume of the tank.

5

SMART GPS TRACKER USING ARDUINO

BY ASHWINI KUMAR SINHA

Parents often worry about their children when they are away from them. Have they safely reached school? Are they alone at home? Or are they at the playground with friends? - these are some of the questions that bothers parents all over the world.

You can watch Video for this DIY here: https://youtu.be/b3N-YZvNAMk

So, today we are going to make a smart tracker that can keep track of a child. Besides this, the device can also be used to track your vehicle location and other objects. Let’s begin.

First, collect the following components.

Components

- GPS Module
- Arduino Nano
- SIM800L
- 2G SIM card
- OLED Display
- Small 3V battery
- Wires

Prerequisites

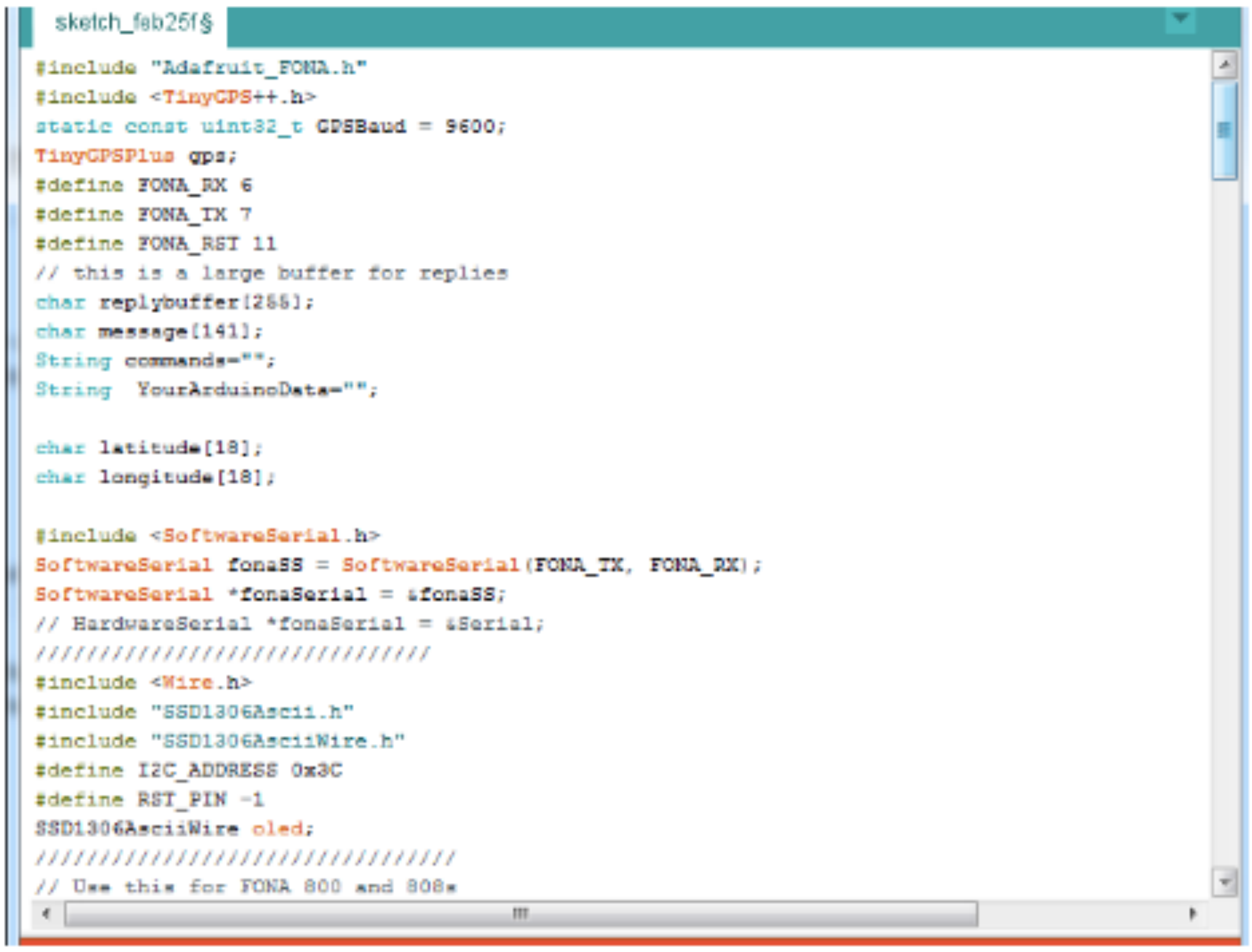
First we need to install the libraries ‘Tiny GPS ++’ and ‘FONA’ in Arduino IDE. To do so, go to tools → click on library manager → search the required library → install the library.

Coding

In first part of the code, initialise the library in code. Next, create some variables that will store the message, GPS location and other data. Then we define the pins for GPS module and SIM800L module.

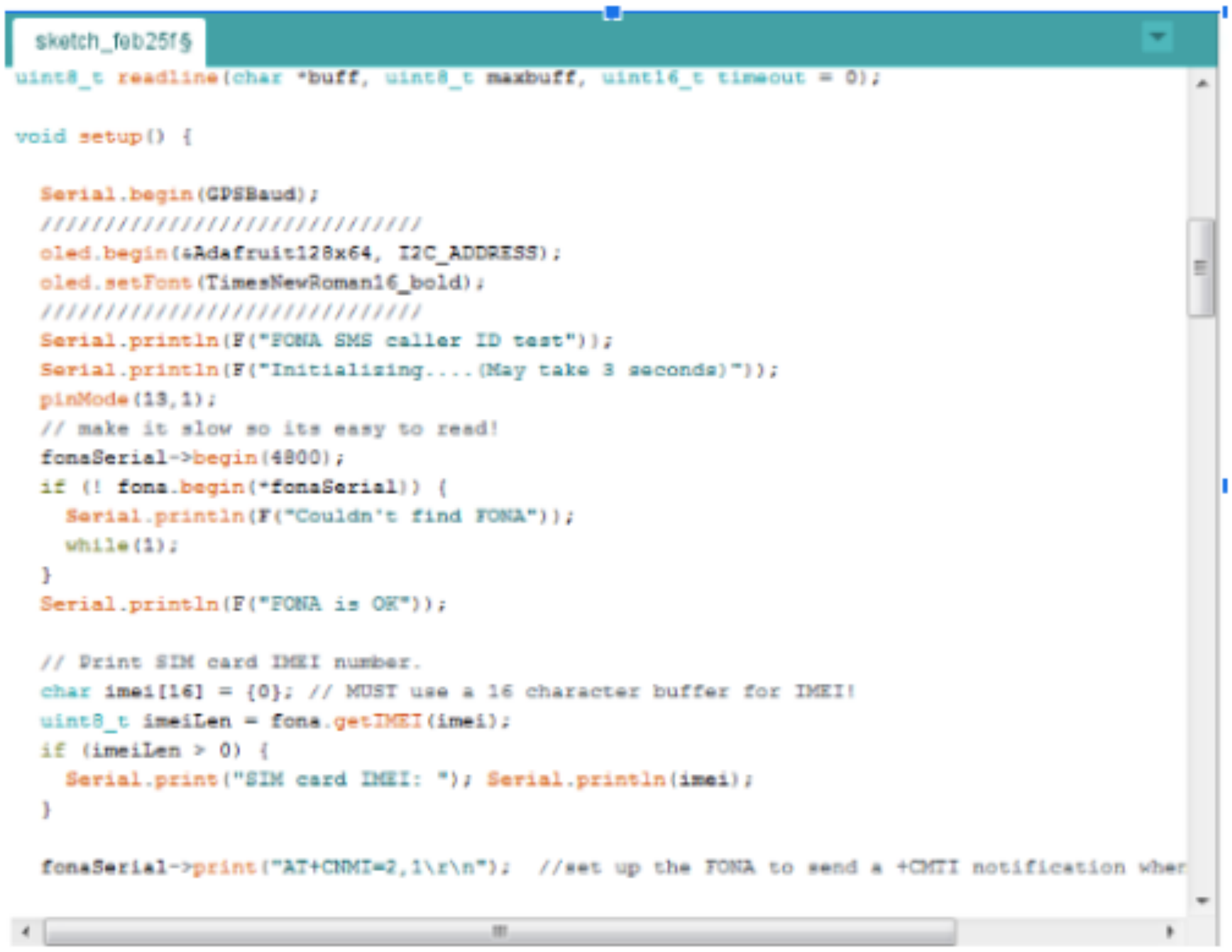
After that, we will create the setup function where the baud rate of SIM800L and GPS modules for serial communication are set to 4800 and 9600 respectively.

Here, we have created a loop function to check whether the incoming message has been received by SIM800L or not. If the message has been received, then convert it into a readable string. It will check the message for a ‘get location’ command, after which the collected data (with respect to location) will be sent back to the original device.



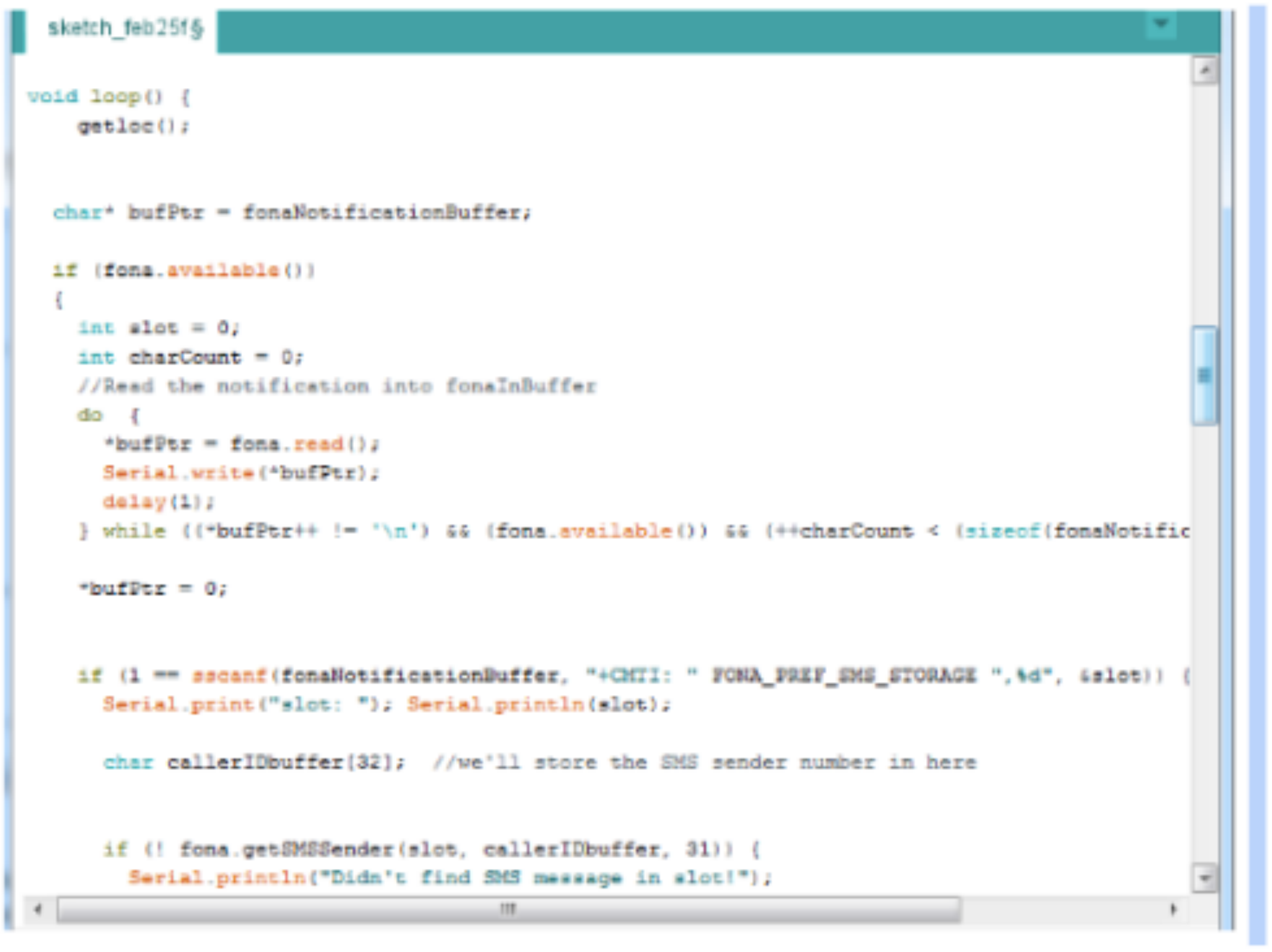
```
#include "Adafruit_FONA.h"
#include <TinyGPS++.h>
static const uint32_t GPSBaud = 9600;
TinyGPSPlus gps;
#define FONA_RX 6
#define FONA_TX 7
#define FONA_RST 11
// this is a large buffer for replies
char replybuffer[255];
char message[141];
String commands="";
String  YourArduinoData="";

char latitude[18];
char longitude[18];

#include <SoftwareSerial.h>
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;
// HardwareSerial *fonaSerial = &Serial;
////////////////////////////////
#include <Wire.h>
#include "SSD1306Ascii.h"
#include "SSD1306AsciiWire.h"
#define I2C_ADDRESS 0x3C
#define RST_PIN -1
SSD1306AsciiWire oled;
//////////////////////////////////
// Use this for FONA 800 and 808s
```

Fig 1.



```
uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);

void setup() {

  Serial.begin(GPSBaud);
  ////////////////////////////////
  oled.begin(&Adafruit128x64, I2C_ADDRESS);
  oled.setFont(TimesNewRoman16_bold);
  ////////////////////////////////
  Serial.println(F("FONA SMS caller ID test"));
  Serial.println(F("Initializing....(May take 3 seconds)"));
  pinMode(13,1);
  // make it slow so its easy to read!
  fonaSerial->begin(4800);
  if (! fona.begin(*fonaSerial)) {
    Serial.println(F("Couldn't find FONA"));
    while(1);
  }
  Serial.println(F("FONA is OK"));

  // Print SIM card IMEI number.
  char imei[16] = {0}; // MUST use a 16 character buffer for IMEI!
  uint8_t imeiLen = fona.getIMEI(imei);
  if (imeiLen > 0) {
    Serial.print("SIM card IMEI: "); Serial.println(imei);
  }

  fonaSerial->print("AT+CNMI=2,1\r\n");  //set up the FONA to send a +CMTI notification wher
```

Fig 2.



```
void loop() {
   getloc();


  char* bufPtr = fonaNotificationBuffer;

  if (fona.available())
  {
    int slot = 0;
    int charCount = 0;
    //Read the notification into fonaInBuffer
    do  {
      *bufPtr = fona.read();
      Serial.write(*bufPtr);
      delay(1);
    } while ((*bufPtr++ != '\n') && (fona.available()) && (++charCount < (sizeof(fonaNotific

    *bufPtr = 0;


    if (1 == sscanf(fonaNotificationBuffer, "+CMTI: " FONA_PREF_SMS_STORAGE ",%d", &slot)) {
      Serial.print("slot: "); Serial.println(slot);

      char callerIDbuffer[32];  //we'll store the SMS sender number in here


      if (! fona.getSMSSender(slot, callerIDbuffer, 31)) {
        Serial.println("Didn't find SMS message in slot!");
```

Fig 3

Within the same loop function, a function named get location() will be called .This function will keep updating the GPS data and return the results to the loop function

Connection

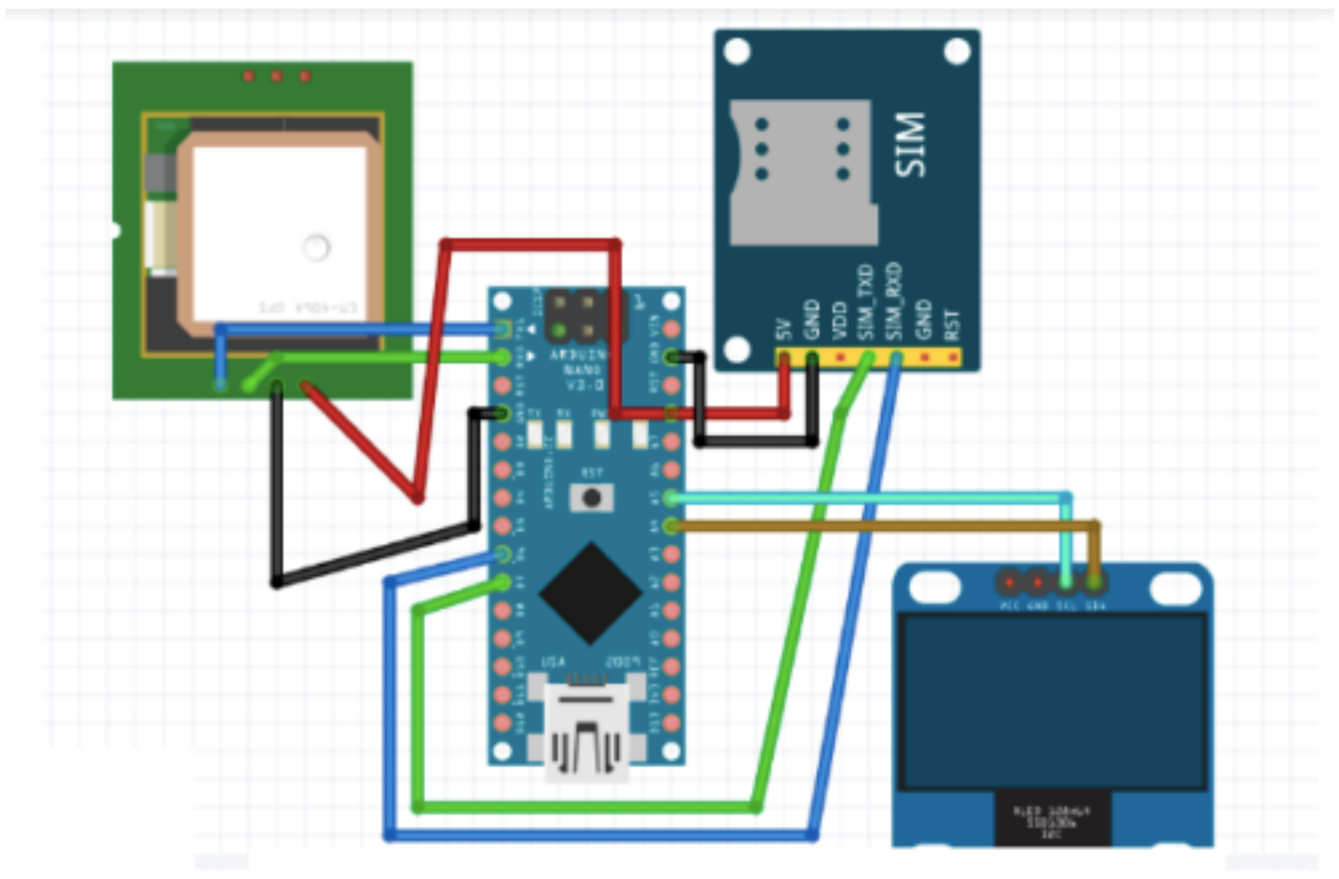


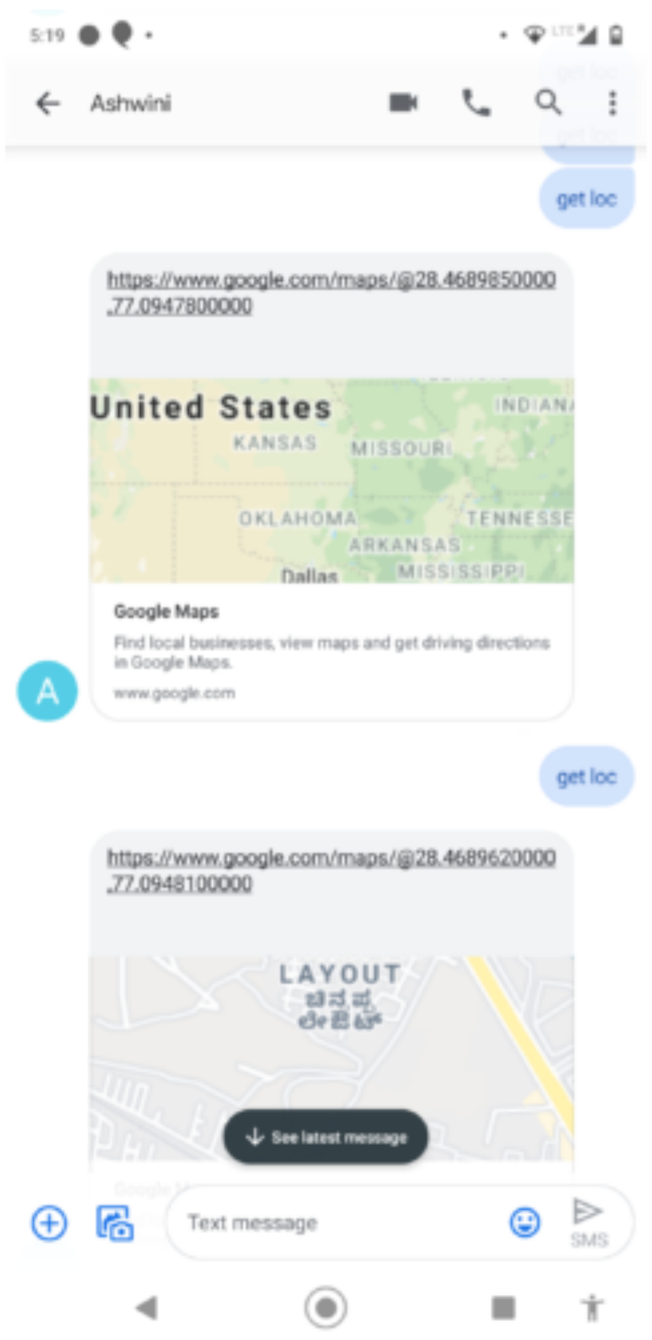
Fig 4.Connection created in Fritzzing

Testing

Power the device with a battery and wait for a few minutes. The device will then become ready for use.

Send the message ‘get location’ to the smart device and you will receive a message with the location URL, which you can check on Google Maps.

Download Source Code



5:19

Ashwini

get loc

https://www.google.com/maps/@28.4689850000,77.0947800000

Google Maps

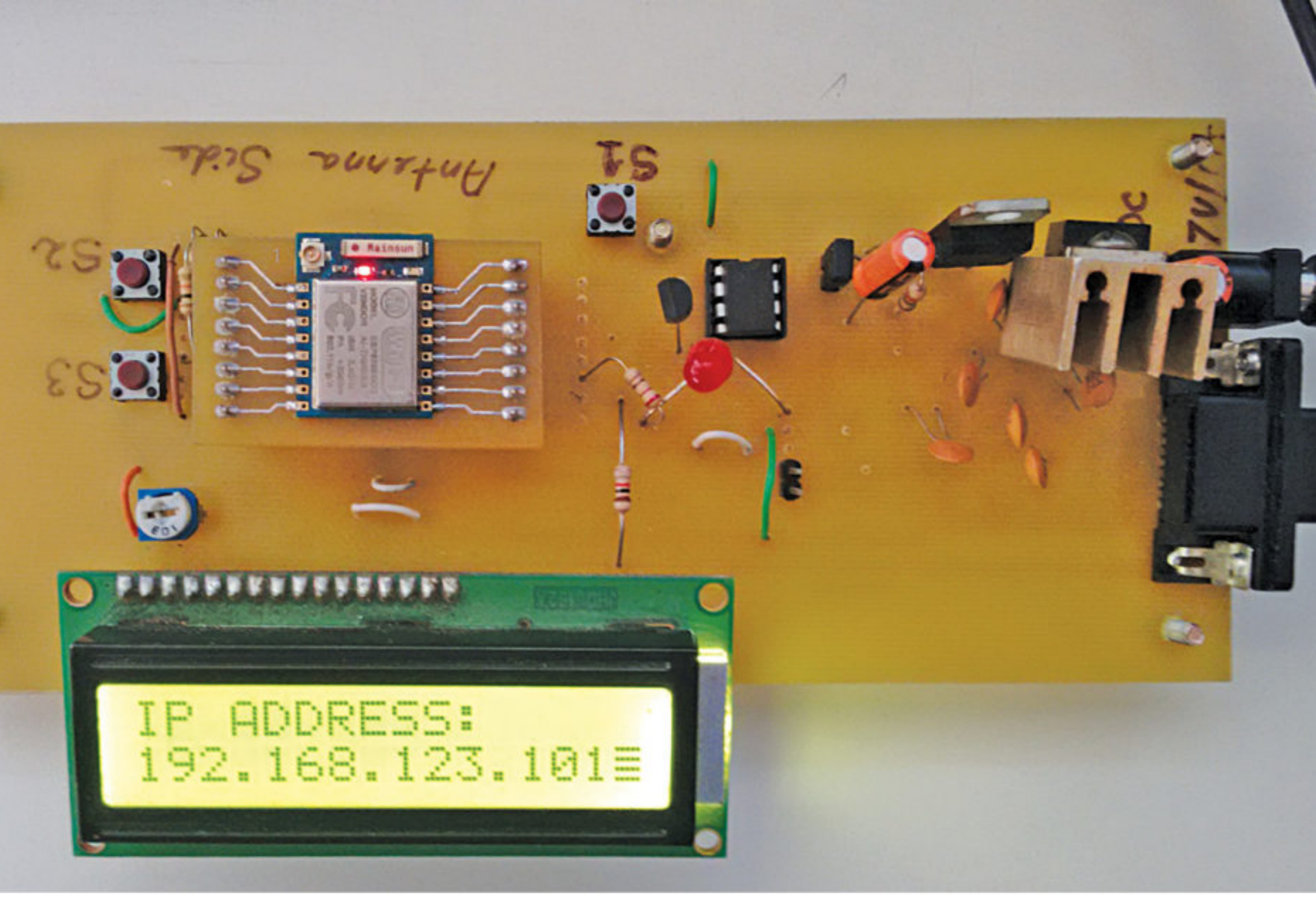
Find local businesses, view maps and get driving directions in Google Maps.

www.google.com

get loc

https://www.google.com/maps/@28.4689620000,77.0948100000

Fig 5.



6

ESP8266 BASED WIRELESS WEB SERVER

BY M. KATHIRESAN

This ESP8266 based wireless web server project is built around an arduino. Currently, ESP8266 is gaining popularity in the field of electronics because it is low-cost, reliable and easily available in the market. Most documents related to this module are in Mandarin (Chinese language) and information provided in the data sheet is not sufficient enough for using ESP8266 for an application.

In order to fill the gap, people from various countries have formed an ESP8266 community forum, which gives necessary details about programming and other related issues concerning this module.

Test Points

Test point	Details
TP1	+5V
TP2	+3.3V

(Voltage measured with respect to ground)

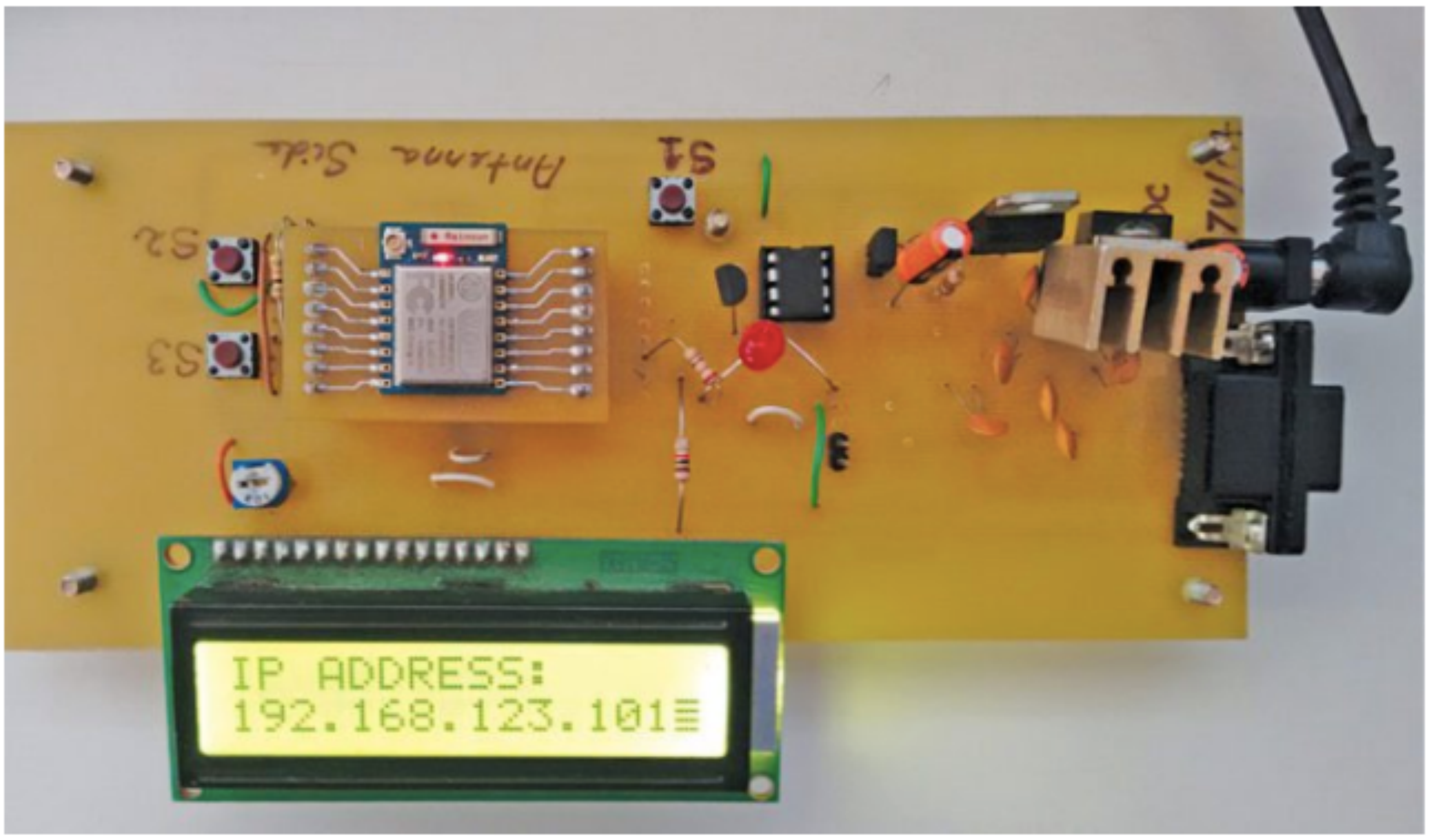
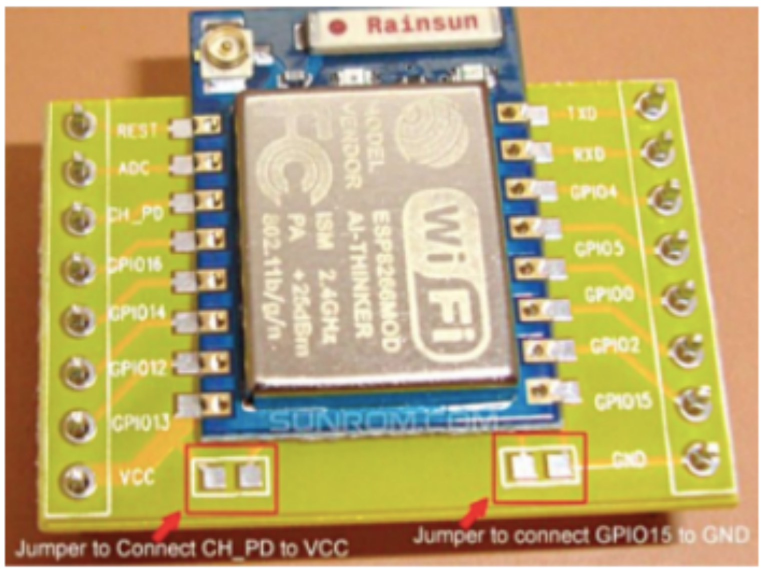


Fig. 1 Author's prototype of the ESP8266 based wireless web server

ESP8266 contains a built-in 32-bit low-power CPU, ROM and RAM. It is a complete and self-contained Wi-Fi network solution that can carry software applications as a stand-alone device or connected with a microcontroller (MCU). The module has built-in AT Command firmware to be used with any MCU via COM port. Salient features of ESP8266 are:

- 802.11 b/g/n protocol
- Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Integrated PLL, regulators and power-management units
- +19.5dBm output power in 802.11b mode
- Supports antenna diversity
- Integrated low-power 32-bit MCU
- SDIO 2.0, SPI, UART

- Wireless SoC
- Has GPIO, I2C, ADC, SPI, PWM
- Maximum frequency is 80MHz
- 64k bytes of instruction RAM
- 96k bytes of data RAM
- 64k bytes of boot RAM
- RISC architecture



ESP8266 module

ESP8266 based wireless web server

Circuit diagrams of the main board and child board are shown in Figs 3 and 4, respectively. We have used ESP8266 as a stand-alone device. Supply voltage for the Wi-Fi module is 3.3 volts. Child board is to be mounted on the main board.

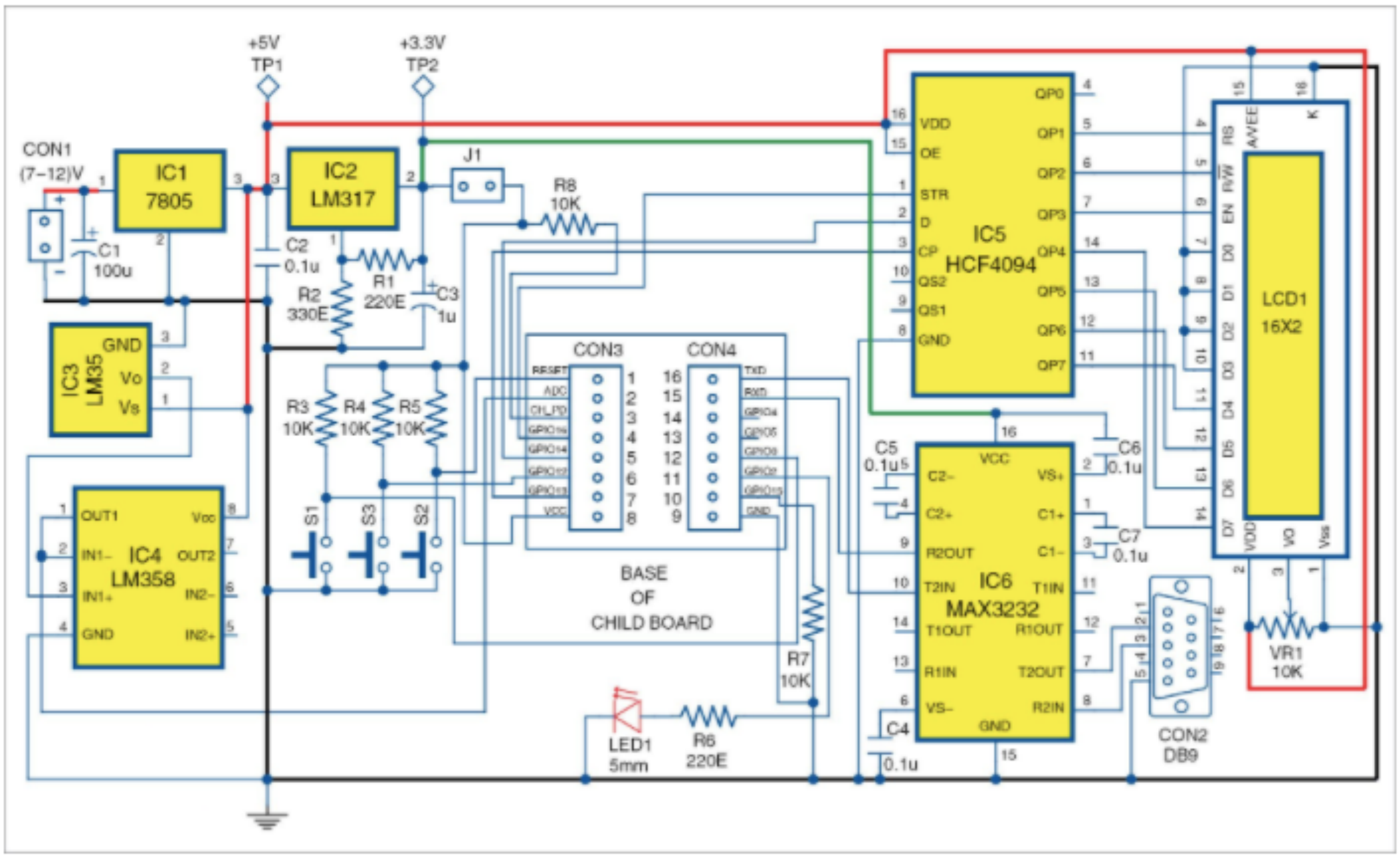


Fig. 3 Circuit diagram of the main board of the low-cost ESP8266 based wireless Web server

CON2 is used for COM port interface for connecting the module to a PC and the module is programmed using a USB-to-serial converter. If physical COM port is available on the PC, USB-to-serial converter is not necessary.

ESP8266 module has 16 pins. Pin 1 (RESET) is connected to 3.3V through resistor R5 and push-button S2 is provided for manual reset. Programing mode pin 12

(GPIO0) is connected to 3.3V through R3, and S1 is used to bring the module to programing mode. Pin 3 (CH_PD) is connected to 3.3V through R8.

ESP8266 also integrates a general-purpose 10-bit resolution ADC (pin 2). It is typically used to measure voltage from the sensor or battery. It cannot be used when the chip is transmitting, otherwise, voltage may be inaccurate.

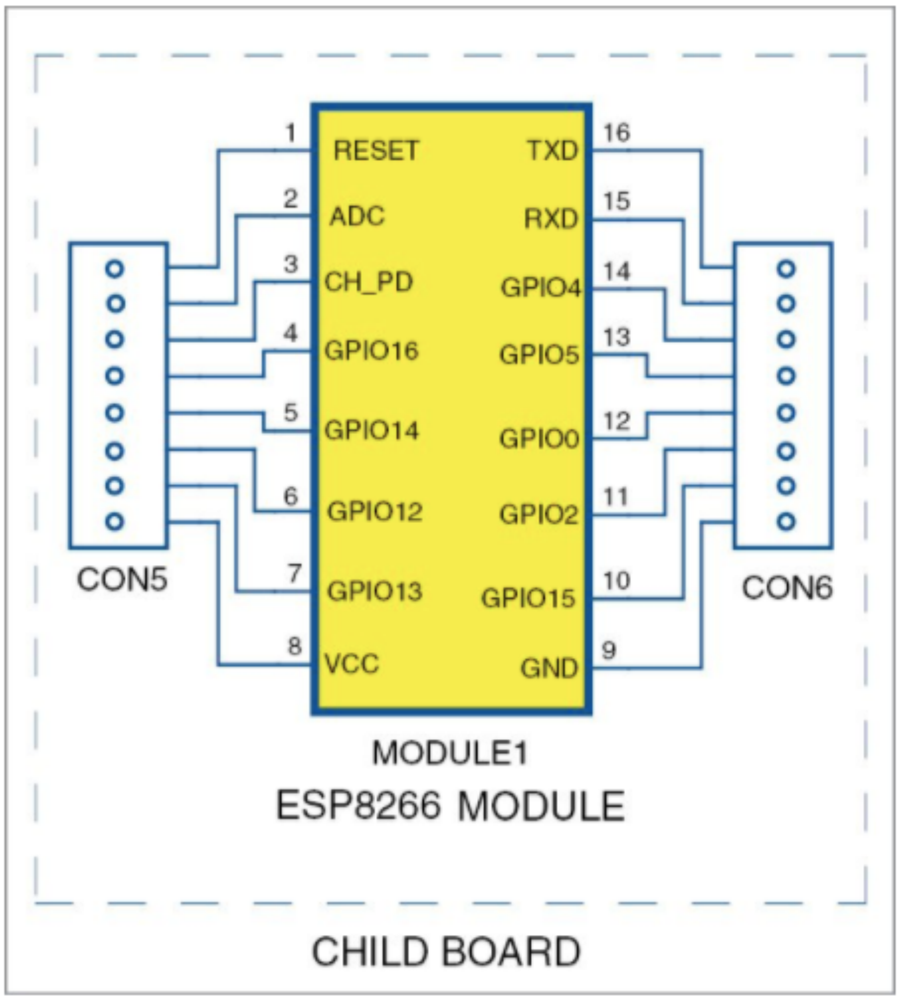


Fig. 4 Circuit diagram of the child board for the circuit shown in Fig. 3

All digital input/output (I/O) pins are protected from over-voltage with a snap-back circuit connected between the pad and the ground. Snap-back voltage is typically 6V and holding voltage is 5.8V. This provides protection from over-voltage and ESD. Output devices are also protected from reverse voltage with diodes. LED1 is connected to pin 11 (GPIO2). Pin 6 (GPIO12) is connected to 3.3V through R4 and provided with a push-button (S3) for debugging purposes.

Every supplier of Wi-Fi modules follows a different pattern for output pin arrangment. So no standard PCB pattern can be followed. Readers of EFY can design their own PCB. Authors of this article have purchased ESP8266 module (Sunrom model no. 4255) from Sunrom Technologies and the PCB is designed to suite their requirement.

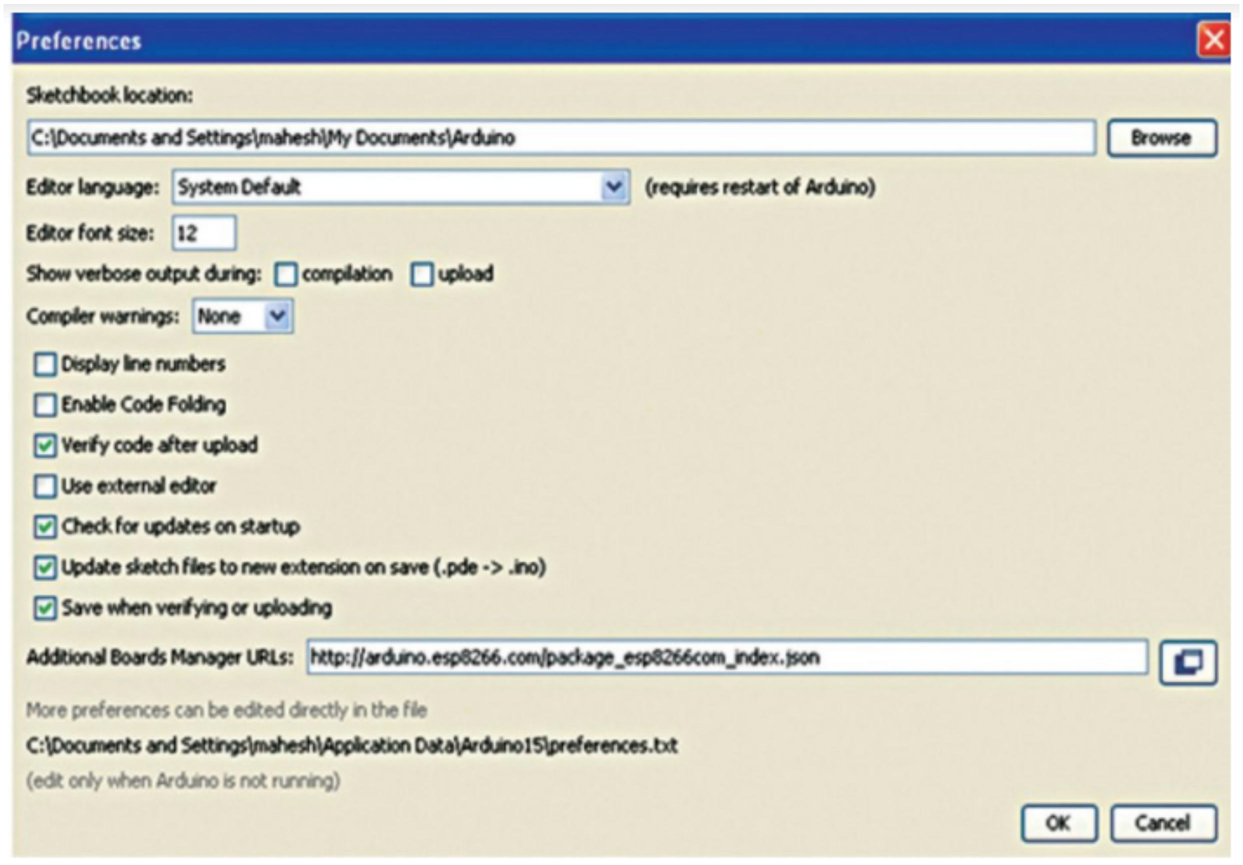
An application has been implemented to measure room temperature using LM35 teperature sensor and an LED control (on/off) using a Web browser. LM35 is a calibrated temperature sensor, whose sensitivity is 10 milli-volt/1°C, but one can build customised projects also.

ESP8266 can be used in home automation, mesh networks, industrial wireless control, IP cameras, sensor networks, smart power plugs, baby monitors, wearable electronics, security ID tags, position system beacons and location-aware devices.

Software

We have used Arduino IDE for compiling and loading programs in this ESP8266 based wireless web server. ESP8266 community has developed a suitable plugin for ESP8266 to use with Arduino IDE. The plugin supports three types of ESP8266 Wi-Fi boards. These are generic ESP8266 board, node MCU ESP8266 board and OLIMEX ESP8266 development board.

Installing the ESP8266 Arduino Addon. Install Arduino IDE (1.6.5 version). Launch Arduino IDE and open preferences window from File ->Preferences option (Fig. 5). Choose Additional Boards Managers URLs: field and enter


Preferences

Sketchbook location:
C:\Documents and Settings\mahesh\My Documents\Arduino
Editor language: System Default (requires restart of Arduino)
Editor font size: 12
Show verbose output during: ☐ compilation ☐ upload
Compiler warnings: None
☐ Display line numbers
☐ Enable Code Folding
☑ Verify code after upload
☐ Use external editor
☑ Check for updates on startup
☑ Update sketch files to new extension on save (.pde -> .ino)
☑ Save when verifying or uploading
Additional Boards Manager URLs: http://arduino.esp8266.com/package_esp8266com_index.json
More preferences can be edited directly in the file
C:\Documents and Settings\mahesh\Application Data\Arduino15\preferences.txt
(edit only when Arduino is not running)

Fig. 5 Preferences window of Arduino IDE

http://Arduino.esp8266.com/stable/package_esp8266com_index.json. Click OK.

Open Boards Manager from Tools -> Board: Arduino Uno -> Boards Manager and scroll down to find ESP8266 by ESP8266 Community. Highlight the field and Install button will appear. Click it. The process will take nearly 30 minutes with an Internet connection.

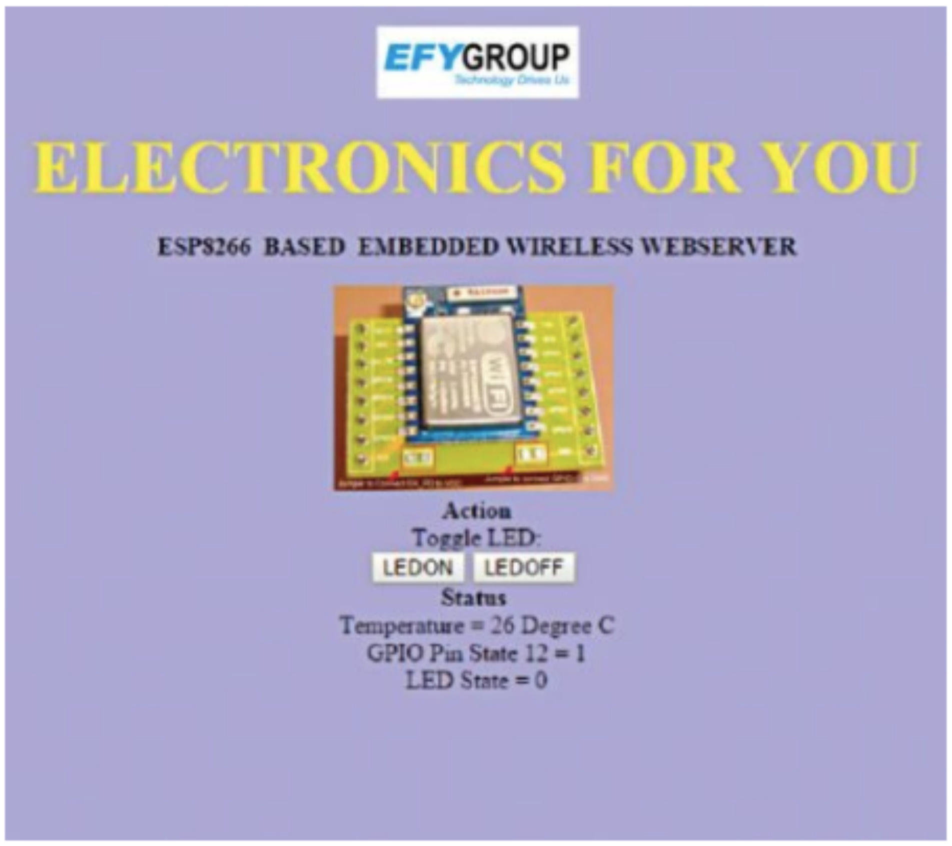
The software is simple. setup() routine initalises the hardware by configuring the serial port, GPIO2 pin as output and GPIO12 as input. setupWiFi() routine configures the wireless section. loop() routine handles requests from the client.

Function size_t sendProgmem(WiFiClient client, const char progmem[], size_t size), sends data to the client. array IN[ ] corresponds to the Web page (Fig. 6), image[

] array corresponds to EFYLOGO (Fig. 7) and ES[ ] array corresponds to the module (Fig. 2).

Arrays are created using file-to-array converter software hexy. It is free and can be downloaded from the Internet. While creating the array, remove image extension jpg for decoding purpose. All jpg files have been included in this article.

Suitable routines are included to change and store SSID and password. After configuration of ESP8266 is done for Wi-Fi, an 8-bit serial shift to parallel-shift register program using HCF4094 is used for LCD interface.


EFYGROUP

ELECTRONICS FOR YOU

ESP8266 BASED EMBEDDED WIRELESS WEBSERVER

Action
Toggle LED:

Status
Temperature = 26 Degree C
GPIO Pin State 12 = 1
LED State = 0

Fig. 6 Web page window

EFYGROUP
Technology Drives Us

Fig.7 EFY logo

Construction and testing

A single-side PCB for the main board is shown in Fig. 8 and its component layout in Fig. 9. An single-side PCB for the child board is shown in Fig. 10 and its component layout in Fig. 11. IC6 to be soldered to the bottom side after proper identification.

Install USB-to-serial UART converter (Fig.12) driver software and perform loopback test.

Select Generic ESP8266 Board Module from Tools Board: Arduino Uno. Arduino IDE will accept your board and you can fuse the program using Upload button. To fuse the program into the module, connect the COM port (CON2) of the main board to your PC through serial-to-USB converter and select as well as note down the emulating virtual COM port number.

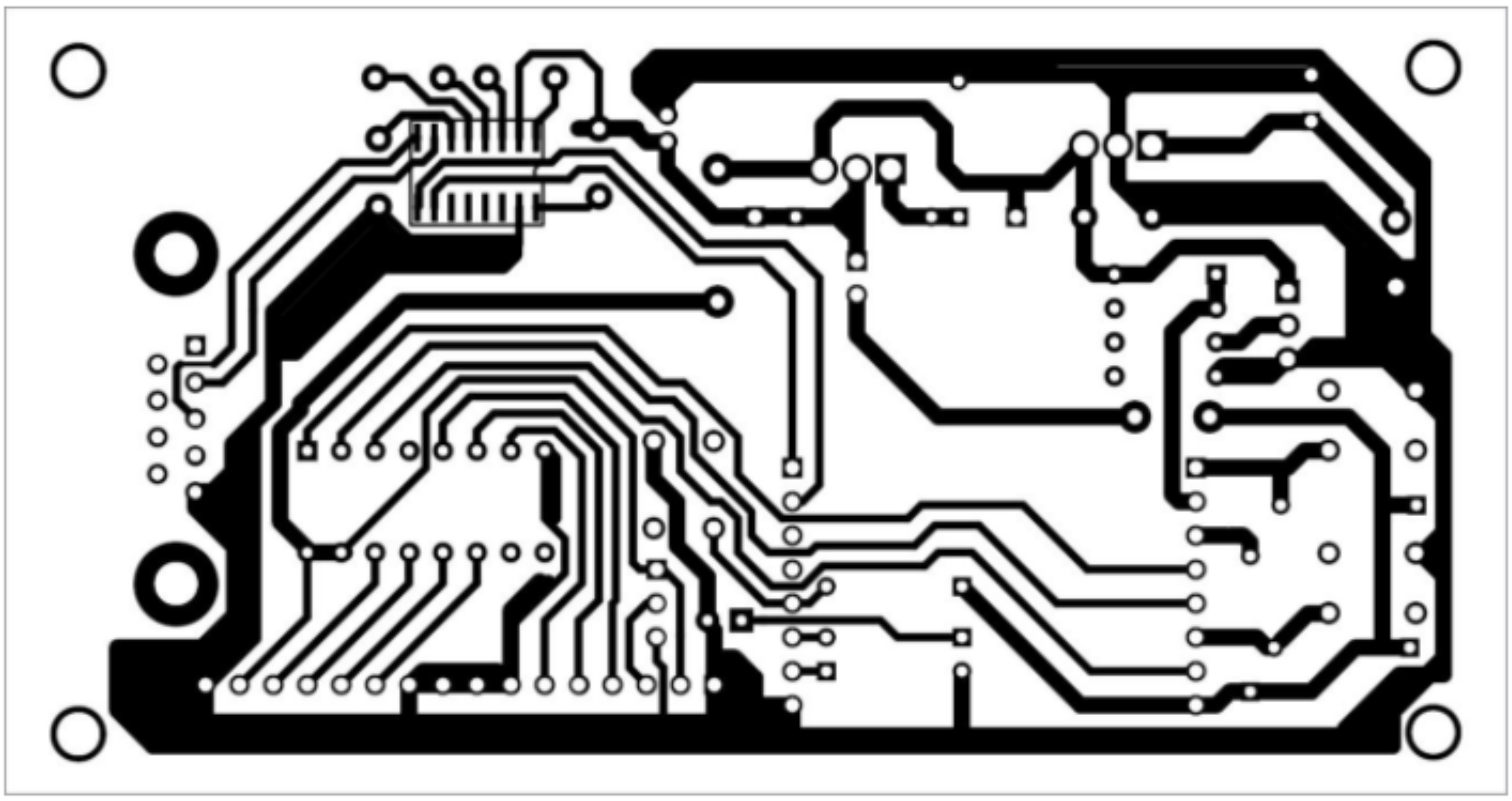


Fig. 8 PCB pattern of the main board

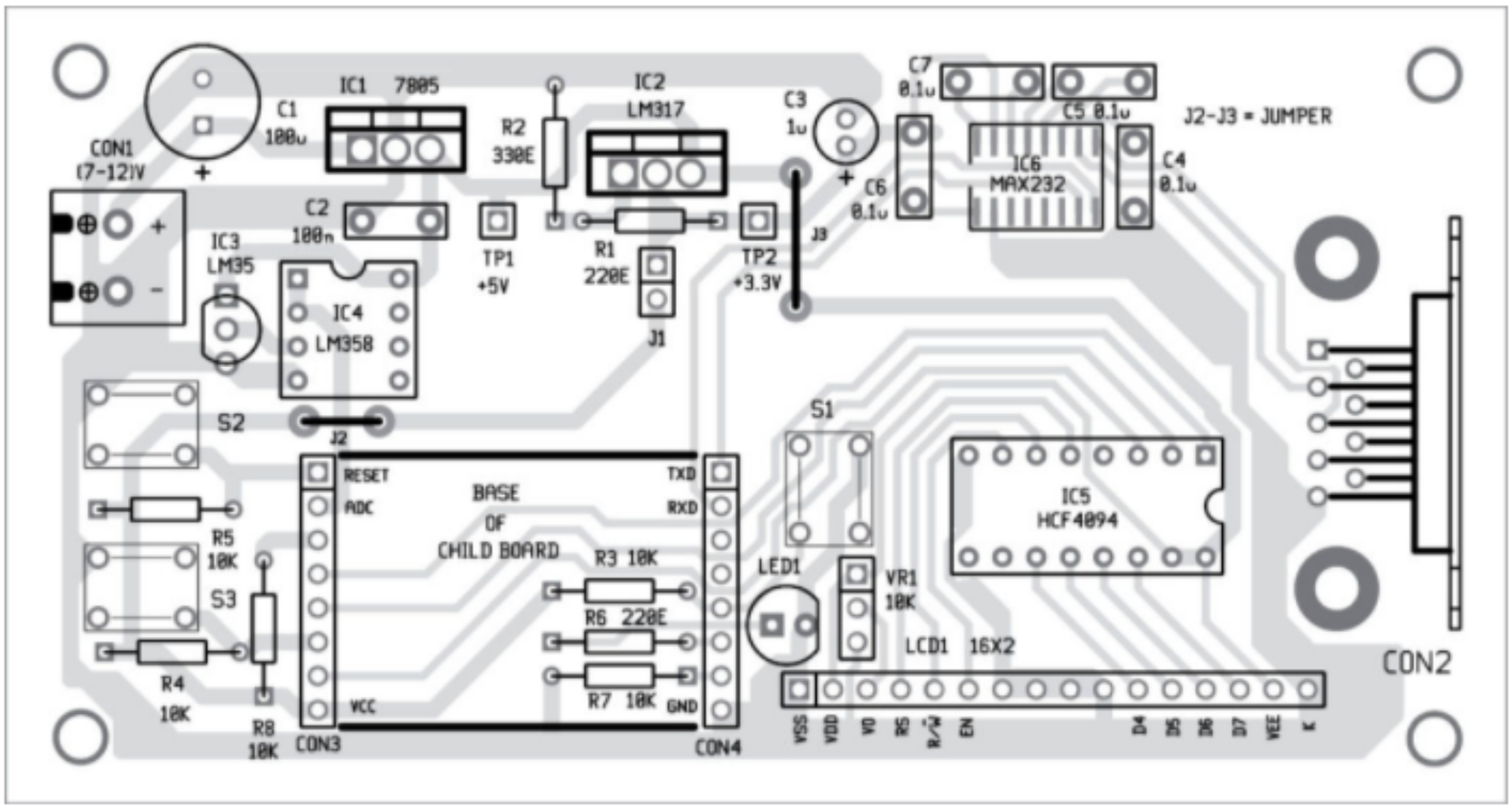


Fig. 9 Component layout of the main board

Power-on the main board

Holding Program button S1 (GPIO0) down, press Reset button (S2) down. Release Reset and then release S1 (GPIO0) button. Now ESP8266 will enter into program mode. Click Upload button and the program will get compiled and generated bin file will load into the module.

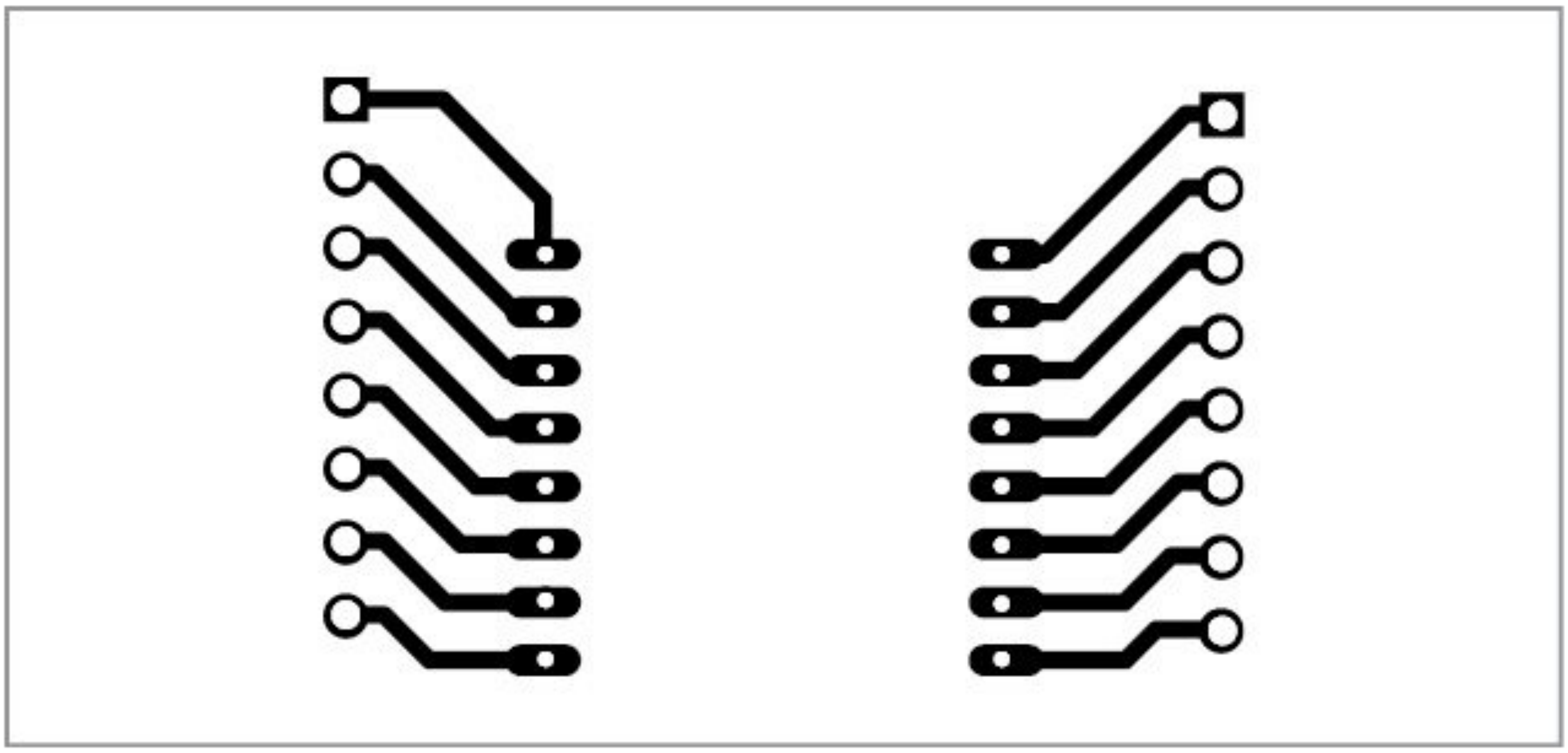


Fig. 10 PCB pattern of the child board

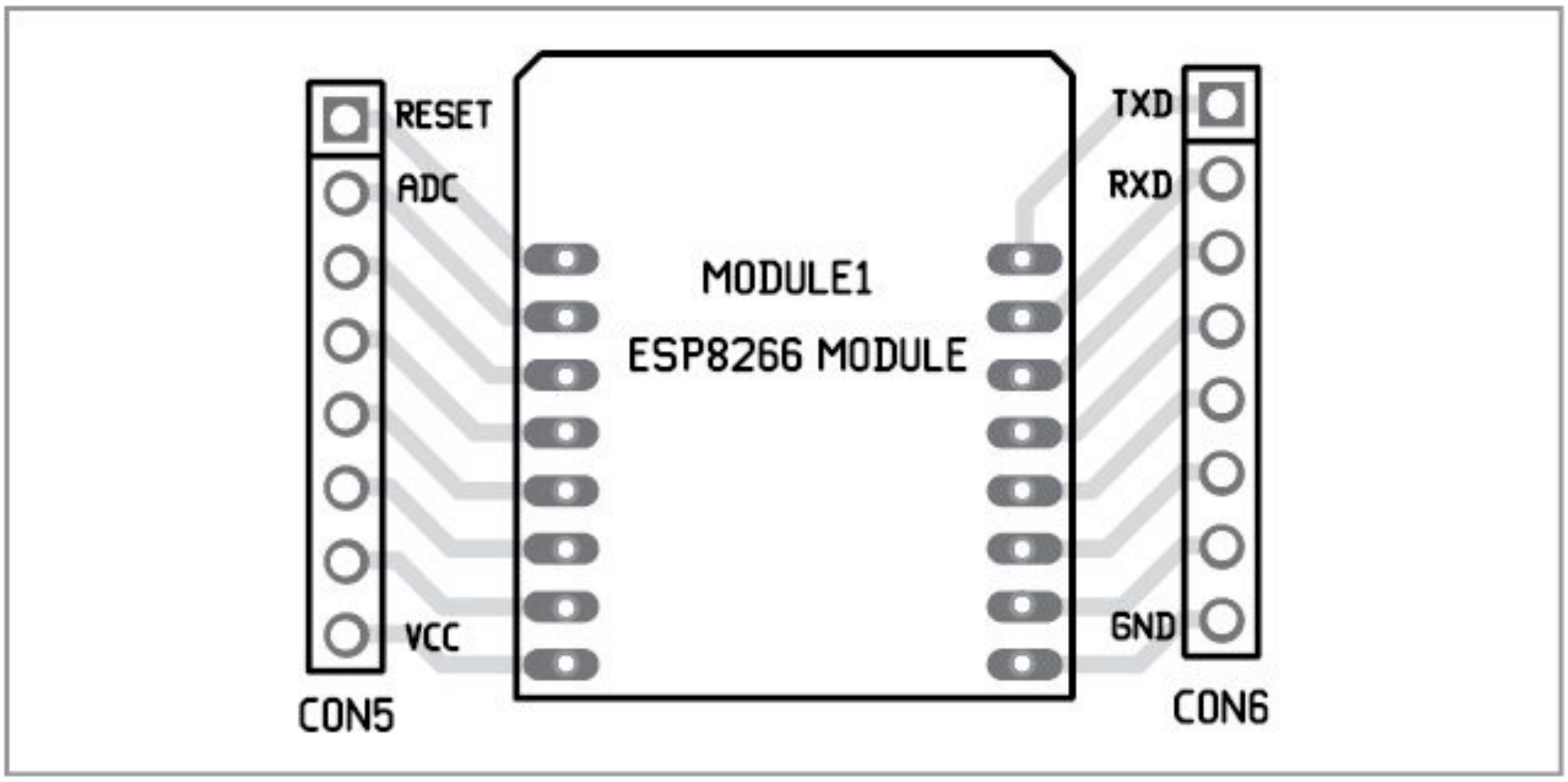


Fig. 11 Component layout of the child board

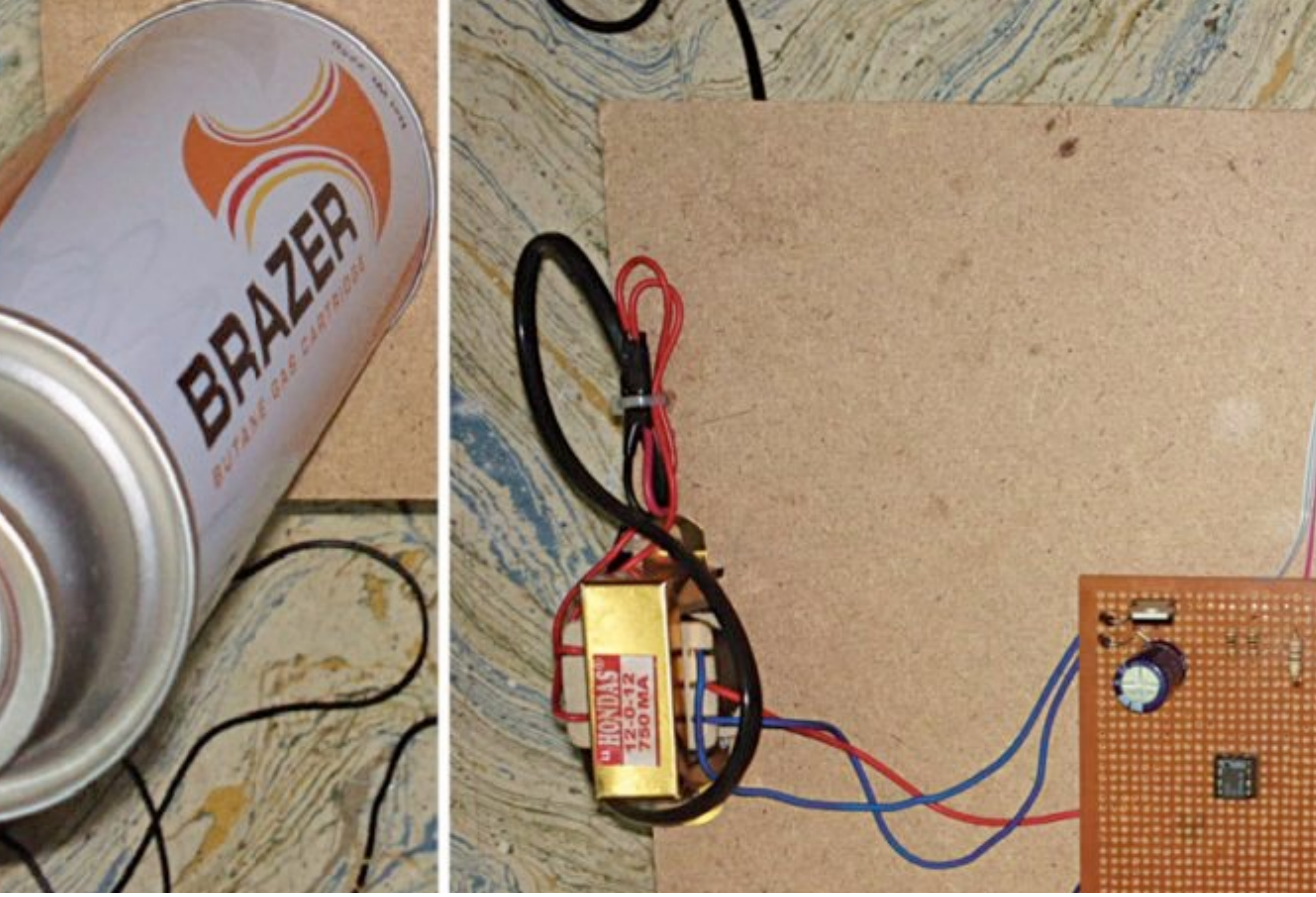
Power-off the main board and power-on your Wi-Fi dongle. Open hyperterminal (X-CTU hyperterminal has been used in EFY Lab) on the PC with baud rate 115200 and select COM port. Hold S3 down and power-on the main board. You will get a prompt for entering SSID and password. Release switch and enter SSID and password of your Wi-Fi network seperated by comma. Click Enter. SSID and password will be stored in the EEPROM.

For subsequent use, you need not use S3. When Wi-Fi is connected, IP address of ESP8266 module will be displayed in LCD1. Launch the browser with the IP address. You will get a Web page as shown in Fig. 6. Buttons marked as LEDON and LEDOFF are for switching on and off LED1. Appliances like fans and motors can be switched on and off with a suitable relay driver circuit using optocouplers connected to pin 11 (GPIO2). Status of pin 6 (GPIO12) and temperature readings are displayed using AJAX scripts without refreshing the entire Web page.

Fig. 12 USB-to-serial UART converter

Download PCB and component layout PDFs: click here

Download source code: click here



7

LOW-COST LPG LEAKAGE DETECTOR

BY PAMARTHI KANAKARAJA

The circuit for an LPG leakage detector is readily available in the market, but it is extremely expensive and usually based on a microcontroller (MCU). Presented here is a low-cost circuit for an LPG detector that you can build easily.

The main objective of the circuit is to detect LPG leakage anywhere. Fig. 1 shows the author's prototype.

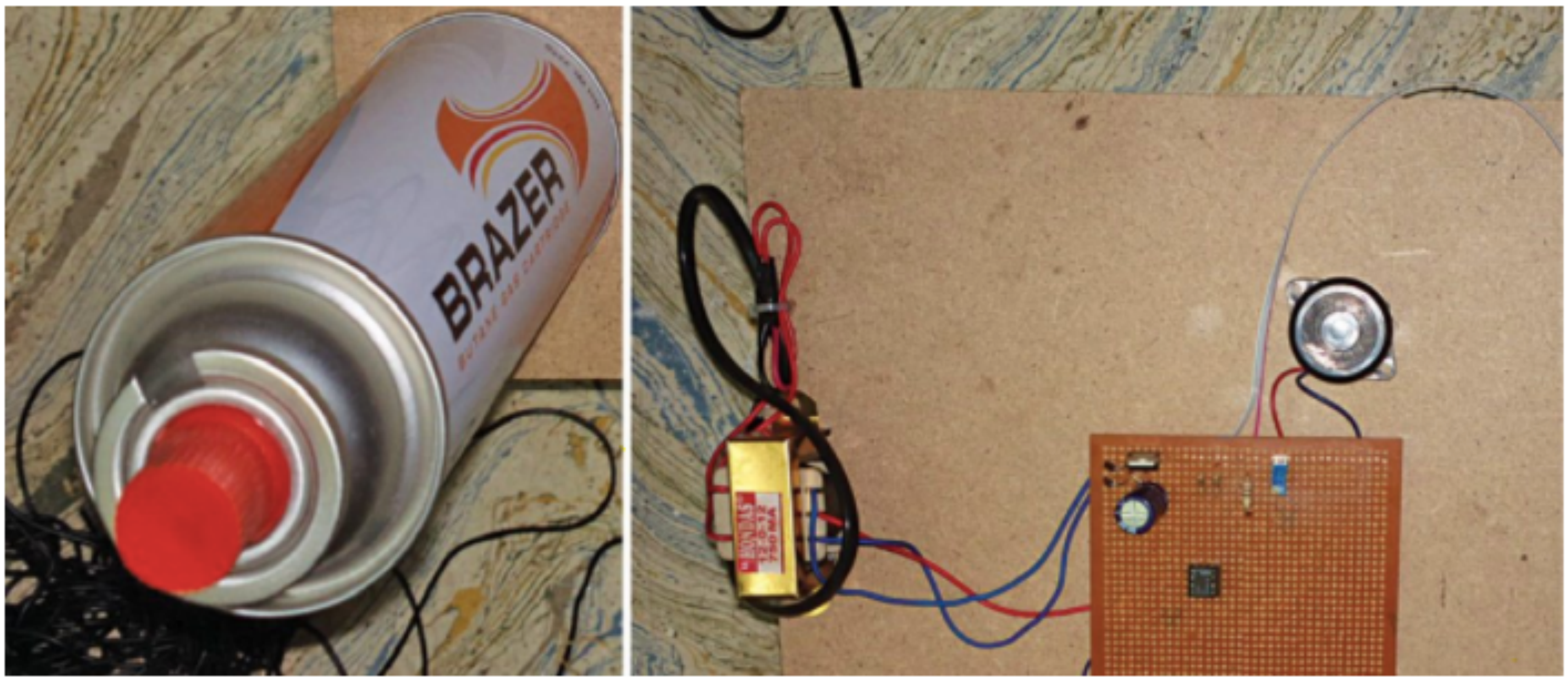


Fig. 1: Author's prototype

Circuit and Working of the LPG leakage detector

Circuit diagram of the low-cost LPG detector is shown in Fig. 2. It is built around step-down transformer X1, two rectifier diodes 1N4007 (D1 and D2), a 1000μF capacitor (C1), 7805 voltage regulator (IC1), MQ-6 LPG gas sensor (GS1), dual comparator LM393 (IC2), darlington transistor TIP122 (T2), 12V high-gain siren/buzzer (PZ1) and a few other components.

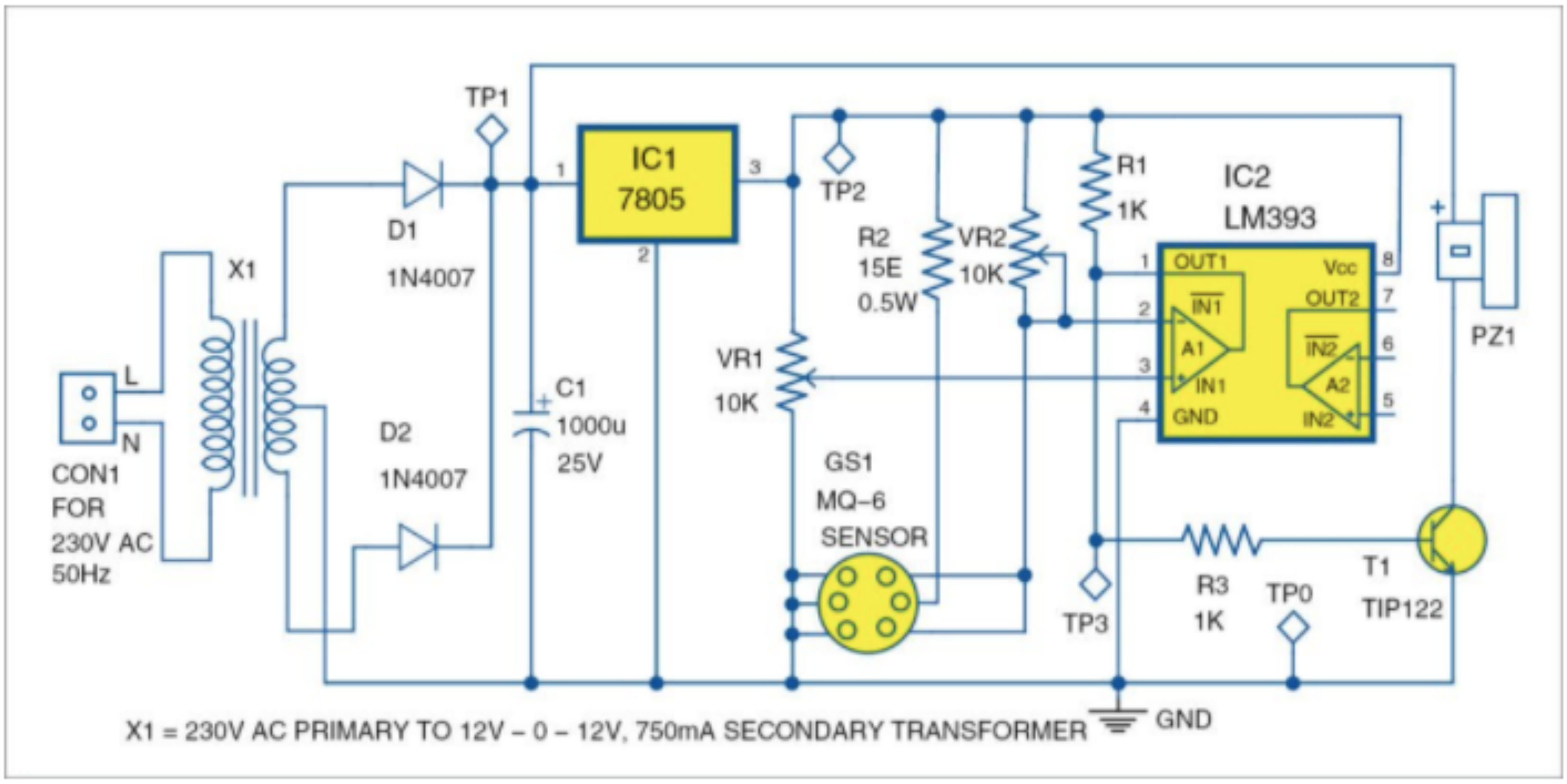
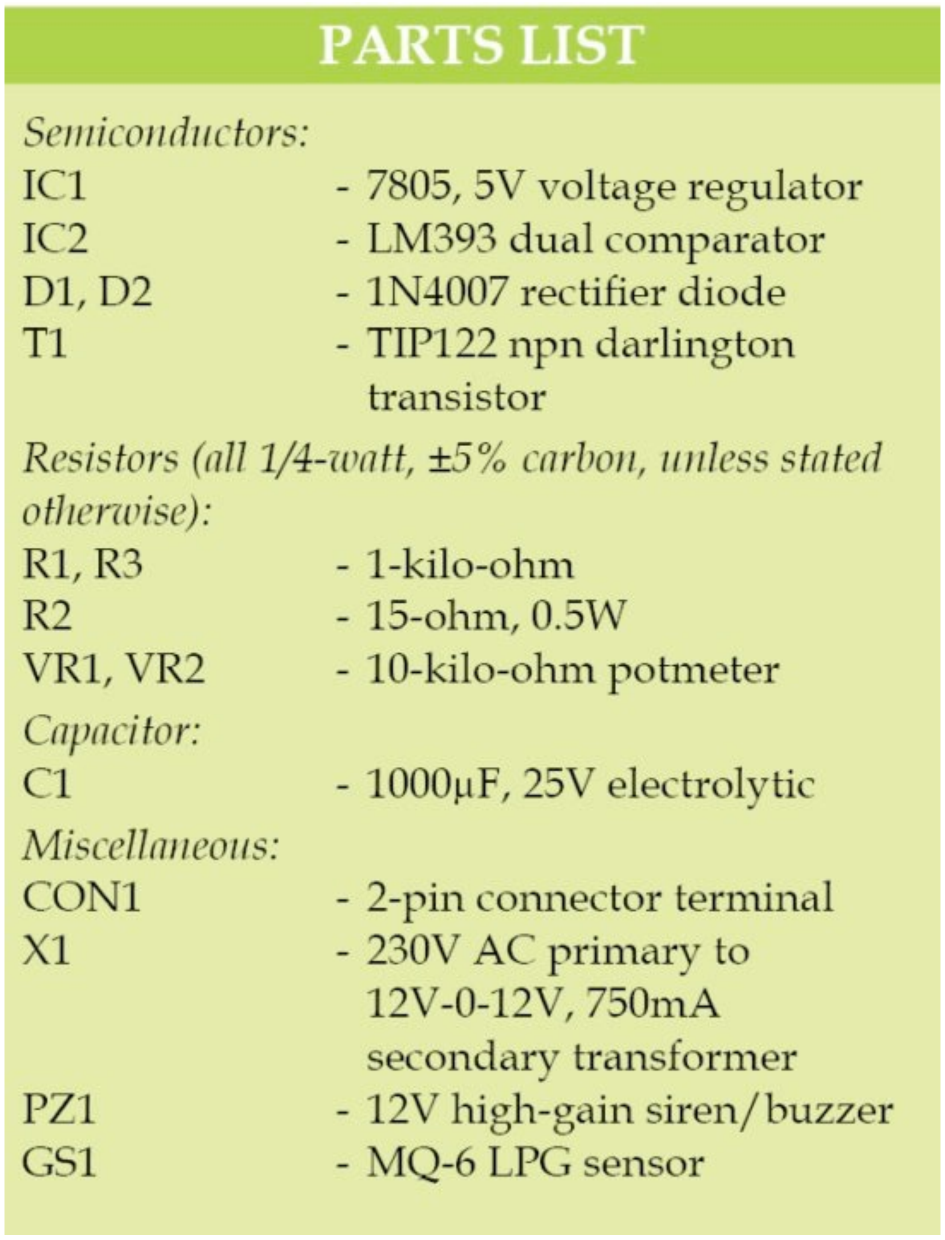


Fig. 2: Circuit diagram of the LPG detector

The mains supply is stepped down by transformer X1, rectified by a full-wave rectifier comprising diodes D1 and D2, filtered by capacitor C1 and fed to regulator 7805 (IC1) to maintain constant 5V DC output, which is fed to the circuit.



PARTS LIST

Semiconductors:	
IC1	- 7805, 5V voltage regulator
IC2	- LM393 dual comparator
D1, D2	- 1N4007 rectifier diode
T1	- TIP122 npn darlington transistor
Resistors (all 1/4-watt, ±5% carbon, unless stated otherwise):	
R1, R3	- 1-kilo-ohm
R2	- 15-ohm, 0.5W
VR1, VR2	- 10-kilo-ohm potmeter
Capacitor:	
C1	- 1000µF, 25V electrolytic
Miscellaneous:	
CON1	- 2-pin connector terminal
X1	- 230V AC primary to 12V-0-12V, 750mA secondary transformer
PZ1	- 12V high-gain siren/buzzer
GS1	- MQ-6 LPG sensor

At the heart of the circuit is dual comparator IC LM393 (IC2). It is used to compare two different voltages, namely, reference voltage and MQ-6 gas sensor output voltage.

Reference voltage at non-inverting pin 3 of IC2 is set using potmeter VR1 to adjust voltage levels based on sensitivity requirements. LPG sensor (MQ-6) output voltage is fed to inverting pin 2 of IC2.

If reference voltage (pin 3 of IC2) is less than sensor voltage (pin 2 of IC2), output goes low, which means there is no LPG leakage. With low output, T1 remains cut-off and there is no current flow through the buzzer; it does not sound and remains in silence mode.

If reference voltage is greater than sensor voltage, output goes high, which means there is LPG leakage. The high output switches on transistor T1 and the buzzer rings loudly to alert the people around.

It is very easy to find gas leakages with this circuit, which uses low-cost components and an interactive way to adjust different sensitivity levels, based on customer needs, with the help of potmeter VR1.

Construction and testing

An PCB pattern of the LPG leakage detector is shown in Fig. 3 and its component layout in Fig. 4.

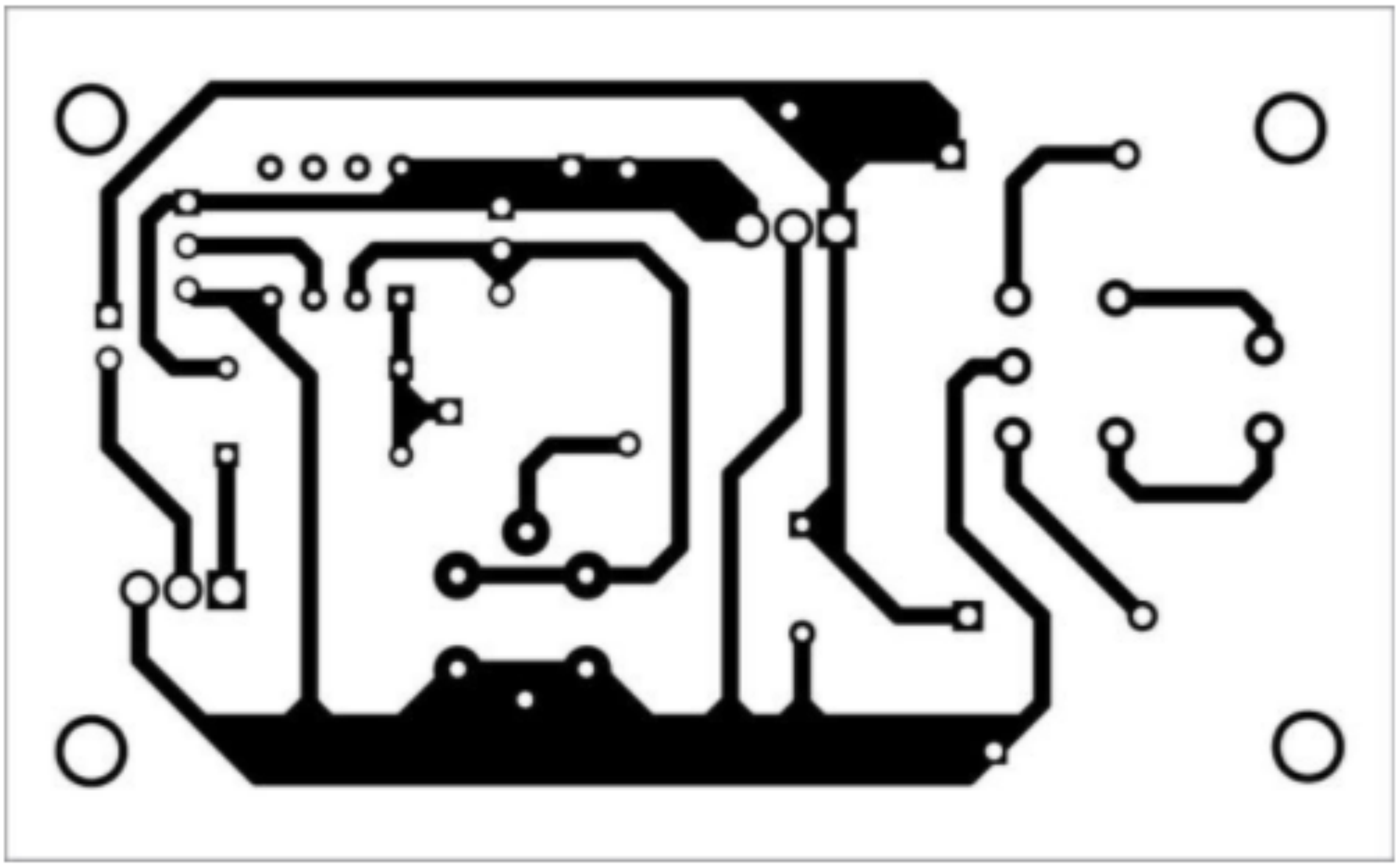


Fig. 3: PCB pattern of the LPG detector

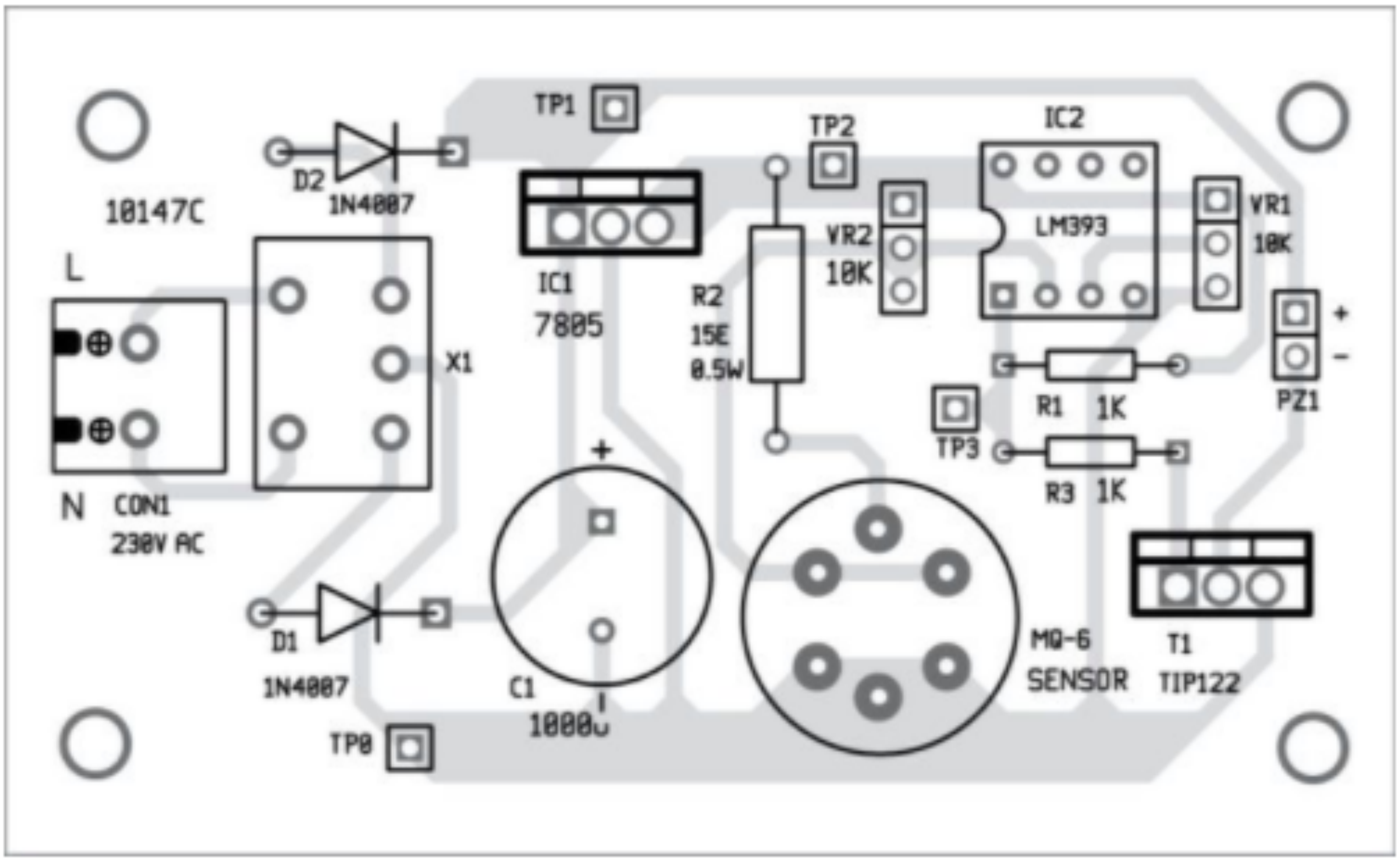


Fig. 4: Component layout of the PCB

Download PCB and component layout PDFs: click here

After assembling the circuit on a PCB, enclose it in a box with an opening for the gas to enter. Place the unit near the LPG cylinder or gas stove, within a distance of one metre. Vary preset VR1 to adjust sensitivity of the sensor.

Verify the voltages are as per test points table before using the circuit. Now, spray the gas from the bottle (as shown on the left side of author's prototype) towards MQ-6 gas sensor and measure voltage at TP3; it should be high.

If you do not have a gas-filled bottle, place the LPG leakage detector near the gas stove burner and turn it on for a few seconds without igniting. Then, turn the burner off and adjust VR1 until buzzer sounds.

8

MOBILE PHONE DETECTOR USING LM358

BY KUMAR ABHISEKH

This mobile phone detector can sense the presence of an activated mobile phone from a distance of four to five metres. So it can come handy in an examination hall or meetings where mobile phones are not permitted.

The circuit can detect incoming and outgoing

PARTS LIST

Semiconductors:	
IC1	- LM358 op-amp
T1	- BC548 npn transistor
LED1	- 5mm LED
Resistors (all 1/4-watt, ±5% carbon):	
R1	- 100-kilo-ohm
R2	- 220-kilo-ohm
R3	- 1-kilo-ohm
VR1	- 2.2-mega-ohm preset
Capacitors:	
C1, C2	- 1µF, 16V electrolytic
C3	- 100µF, 16V electrolytic
Miscellaneous:	
CON1	- 2-pin terminal connector
S1	- On/off switch
ANT.1	- 15cm single-strand wire antenna
	- 4.5V DC power supply

calls, SMSes, Internet and video transmissions even if a mobile phone is kept in silent mode. When it detects an RF signal from an activated mobile phone, its LED starts blinking and continues to blink until the signal stops. The author's prototype is shown in Fig. 1.

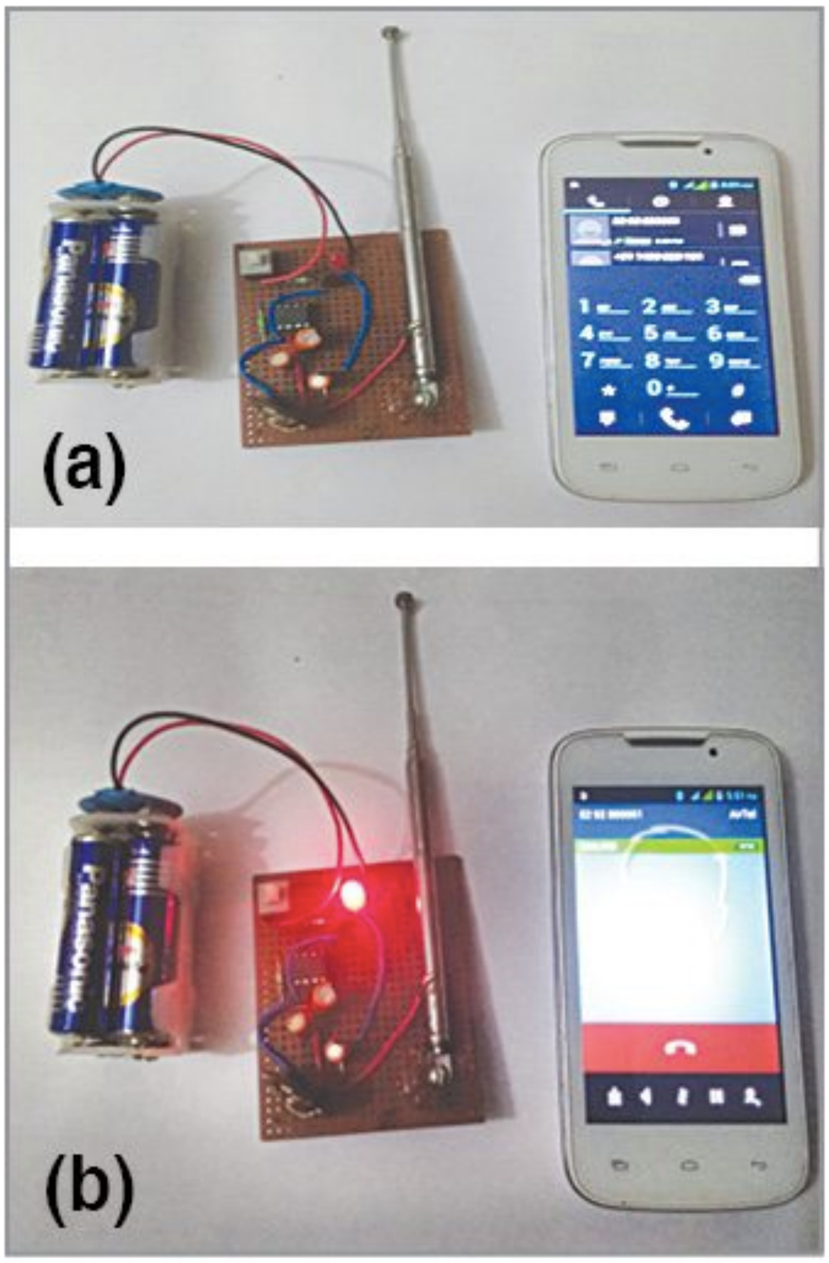


Fig. 1: Author's prototype (a) when a call is not detected and (b) when a call is detected

Circuit and working

Circuit diagram of the mobile phone detector using LM358 is shown in Fig. 2. It is built around LM358 (IC1) and npn transistor BC548 (T1).

When a mobile phone is active, it radiates RF signal that passes through nearby space. The signal contains electromagnetic RF radiation from the phone.

Capacitor C1 is used in the circuit to detect the RF signal from the mobile phone. When the mobile phone radiates energy in the form of RF signal, C1 absorbs it and passes on to the inputs of IC1. This is indicated by the flashing of LED1. Preset VR1 (2.2M) is used to vary the range of the circuit. Transistor T1 is used to amplify the signal obtained at pin 1 of IC1.

The circuit is applicable for 2G networks, GPRS and network search (manual/automatic). It does not detect 3G, WCDMA and HSDPA network signals so well.

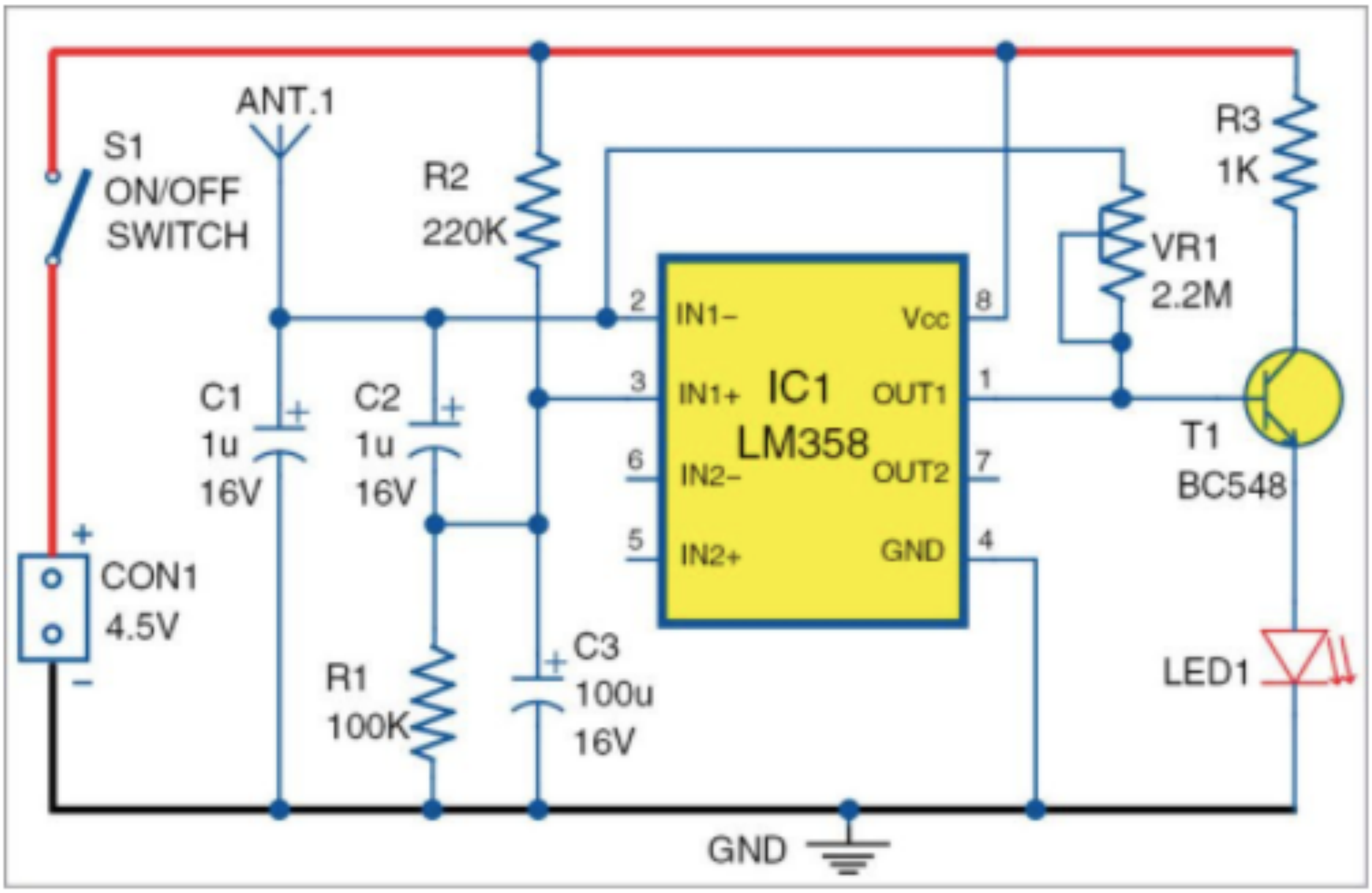


Fig. 2: Circuit diagram of the mobile phone detector

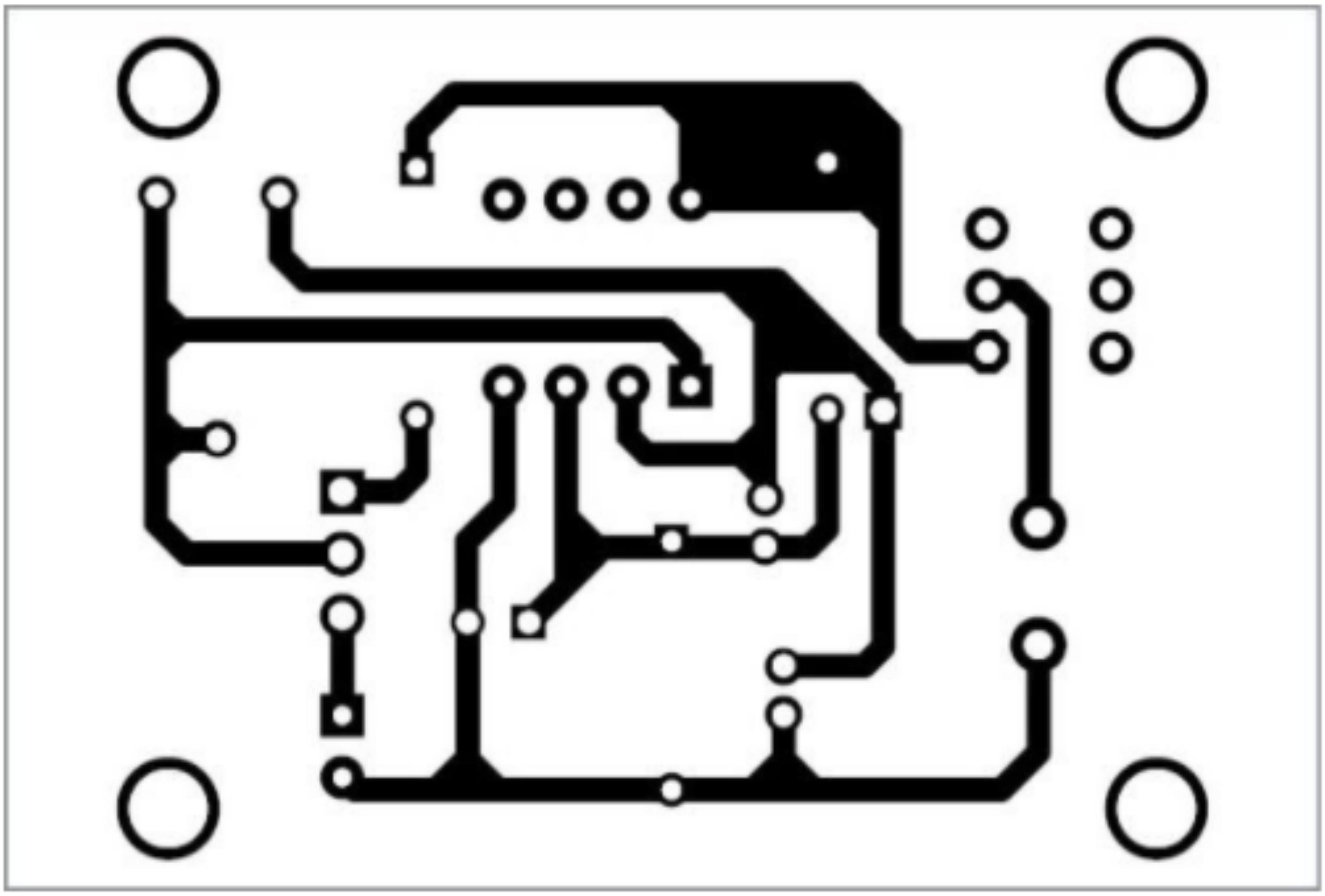


Fig. 3: Actual-size PCB layout for the mobile phone detector circuit

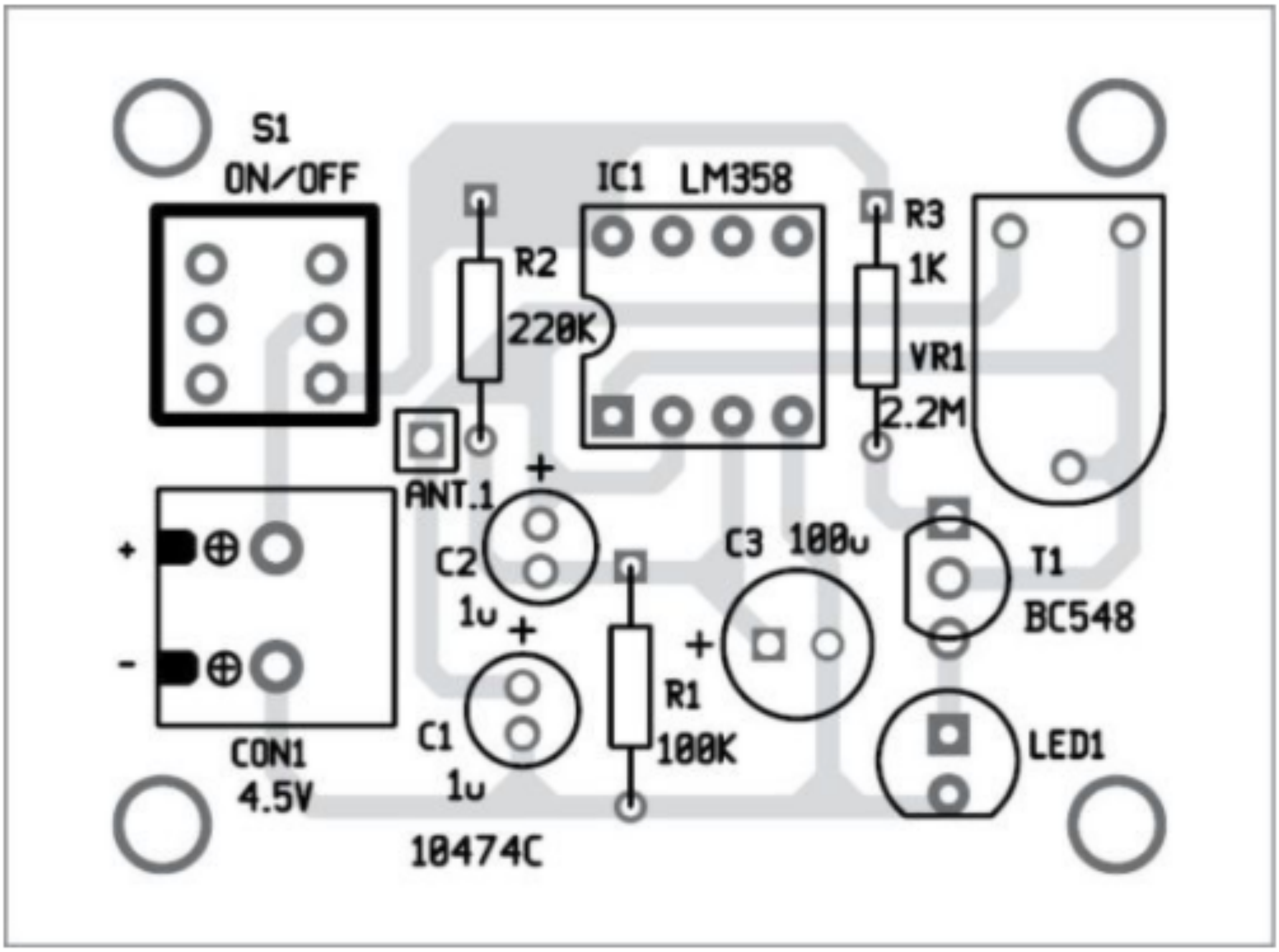


Fig. 4: Component layout of the PCB

Construction and testing

A single-side PCB layout for the mobile phone detector circuit is shown in Fig. 3 and its component layout in Fig. 4. After assembling the circuit on the PCB, enclose it in a suitable plastic box.

Download PCB and component layout PDFs: click here

The circuit works off a 4.5V DC power supply.

9

OTP BASED SMART WIRELESS LOCKING SYSTEM USING ARDUINO

BY ASHWINI KUMAR SINHA

In this project, we are going to make a smart OTP-based locking system as you can see in (Fig. 1). This smart lock can generate a new password every time you unlock it, which further enhances your security level. This new device is much safer than the traditional key-based system and electronic wireless lock system. If you are still using the key-based system, you are likely to land in a big problem if your key gets lost or stolen. The electronic wireless lock system is not safe either. You might forget the password and there is also a high risk being hacked.

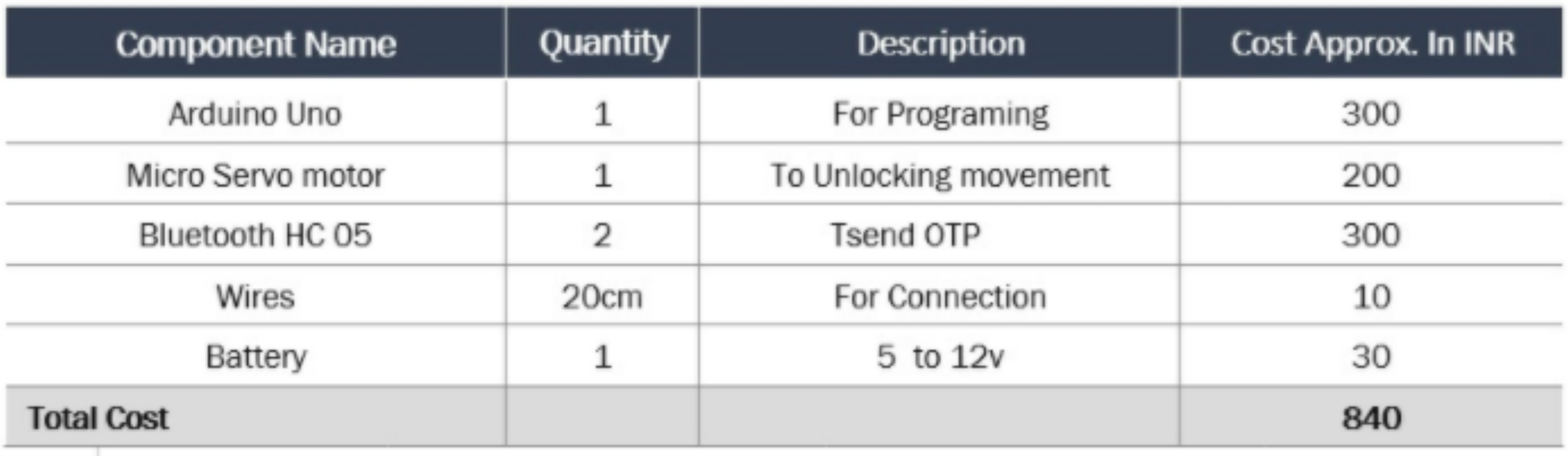
For your safety and security, we bring to you a DIY smart lock that has the capability to remove all these security threats and problems.

Before starting this project, we need to get these materials ready first.

Components Required

- Arduino UNO
- Bluetooth Hc05
- LED
- Some Wires/Jumper Wires
- Servo Motor
- 5V Battery/Power Bank
- USB Type B for Programming Arduino

Bill of materials



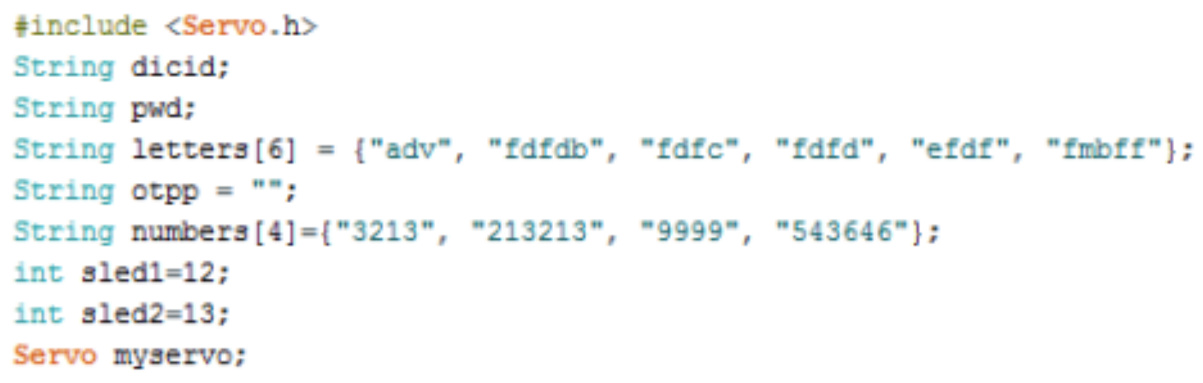
Component Name	Quantity	Description	Cost Approx. In INR
Arduino Uno	1	For Programing	300
Micro Servo motor	1	To Unlocking movement	200
Bluetooth HC 05	2	Tsend OTP	300
Wires	20cm	For Connection	10
Battery	1	5 to 12v	30
Total Cost			**840**

As we have all the components now, we can start our project.

Coding

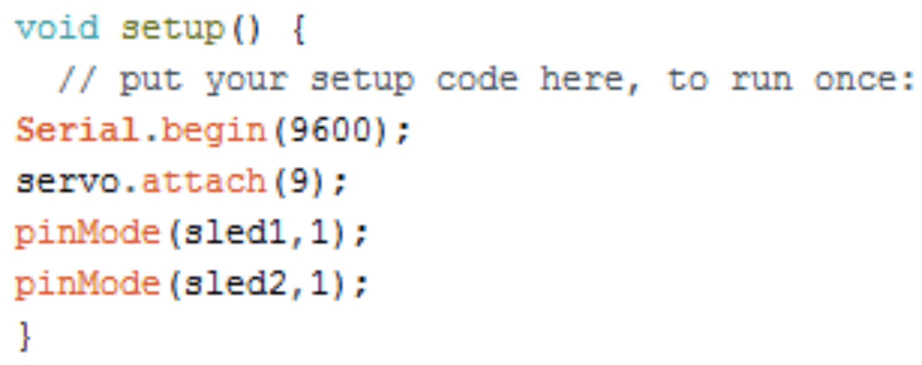
First, we need to include the library and declare variables needed, as in snippet of code. We include a servo library, then create a string array to generate a password. After this, we need to create a few more string variables to store password, OTP and LED pin numbers as in snippet Fig.1 of code.

In the second part of coding, we need to set up serial and Braud rate for Bluetooth. Here I have used 9600 Braud rate but if it didn't work, you can use default Braud rate of Hc 05 i.e (38400). Then, we have to set up a pin for servo using servo.attach (PWM pin number). After that, we can define pin mode as output for led refer (Fig.2)



```
#include <Servo.h>
String dicid;
String pwd;
String letters[6] = {"adv", "fdfdb", "fdfc", "fdfd", "efdf", "fmbff"};
String otpp = "";
String numbers[4]={"3213", "213213", "9999", "543646"};
int sled1=12;
int sled2=13;
Servo myservo;
```

Fig 1. Arduino Code



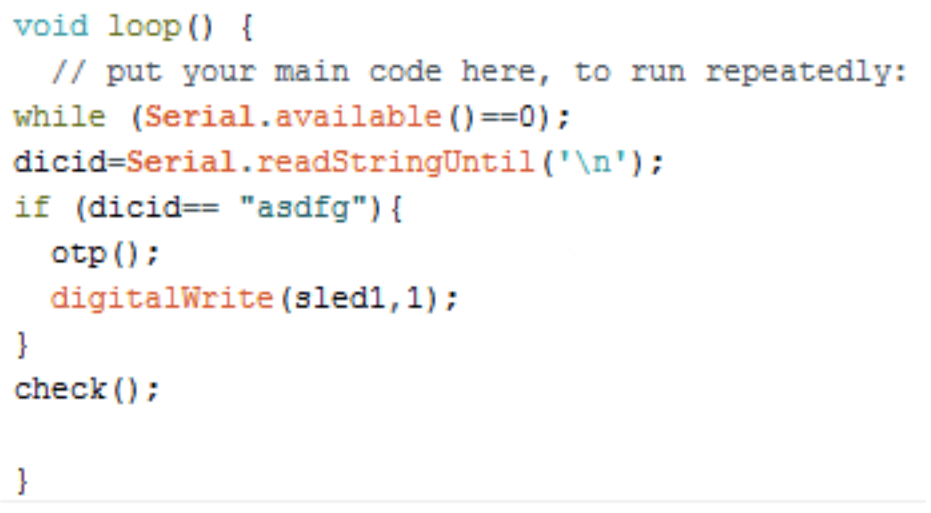
```
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);
servo.attach(9);
pinMode(sled1,1);
pinMode(sled2,1);
}
```

Fig 2. Setting I/O and Bluetooth

In the third part of code, we will create a loop and check the data coming from Bluetooth.Then we create if() statement to check device id. If it matches, then it calls otp() function for generation of OTP(refer Fig. 3)

Then we need to create check() function to check whether OTP is correct or not. If it is correct then it turns servo to open position (refer Fig. 4).

Congrats! We have completed the coding part.

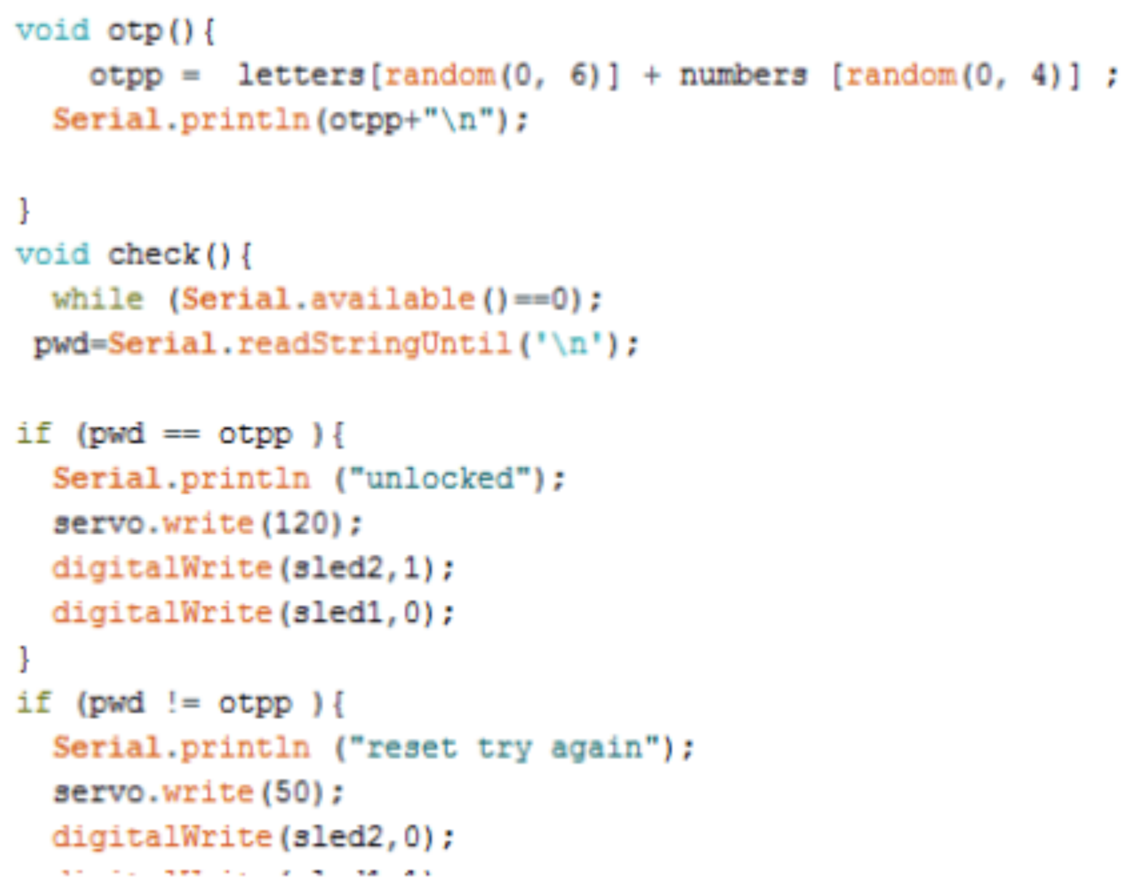


```
void loop() {
  // put your main code here, to run repeatedly:
while (Serial.available()==0);
dicid=Serial.readStringUntil('\n');
if (dicid== "asdfg"){
  otp();
  digitalWrite(sled1,1);
}
check();

}
```

Fig. 3 Arduino code checking device Id

Now, let's make Android APP for our project. We can make App with two different platforms, either Android Studio or MIT app inventor. For this project, let's choose the MIT app inventor because it's easy to make an App with blocks without coding. Comment below if you want me to build App with Android Studio.

APP Building:



```
void otp(){
    otpp =  letters[random(0, 6)] + numbers [random(0, 4)] ;
  Serial.println(otpp+"\n");

}
void check(){
  while (Serial.available()==0);
 pwd=Serial.readStringUntil('\n');

if (pwd == otpp ){
  Serial.println ("unlocked");
  servo.write(120);
  digitalWrite(sled2,1);
  digitalWrite(sled1,0);
}
if (pwd != otpp ){
  Serial.println ("reset try again");
  servo.write(50);
  digitalWrite(sled2,0);
```

Fig. 4 Arduino code for checking OTP

First, we need to create a layout as in the pic (Fig. 5). Just drag and drop or download the .aia file for it from the link below.

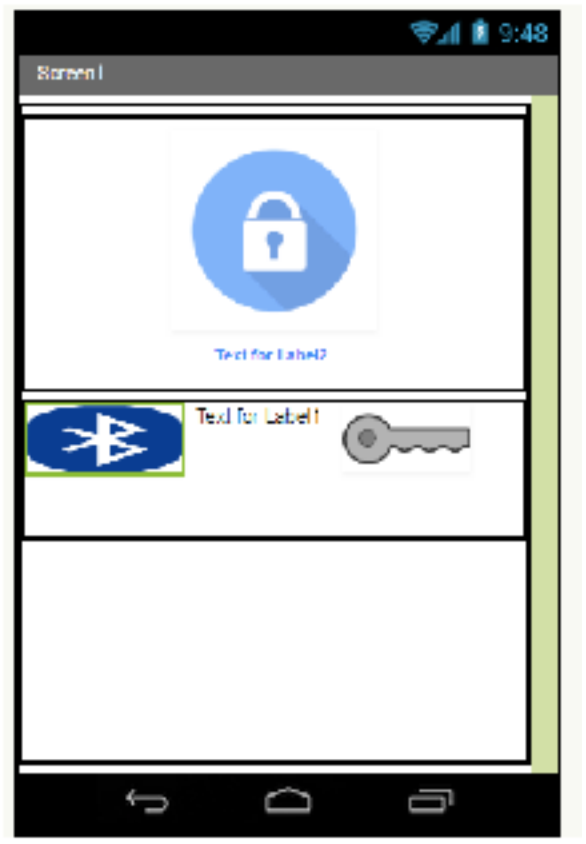


Fig. 5 Layout of App in MIT app inventor

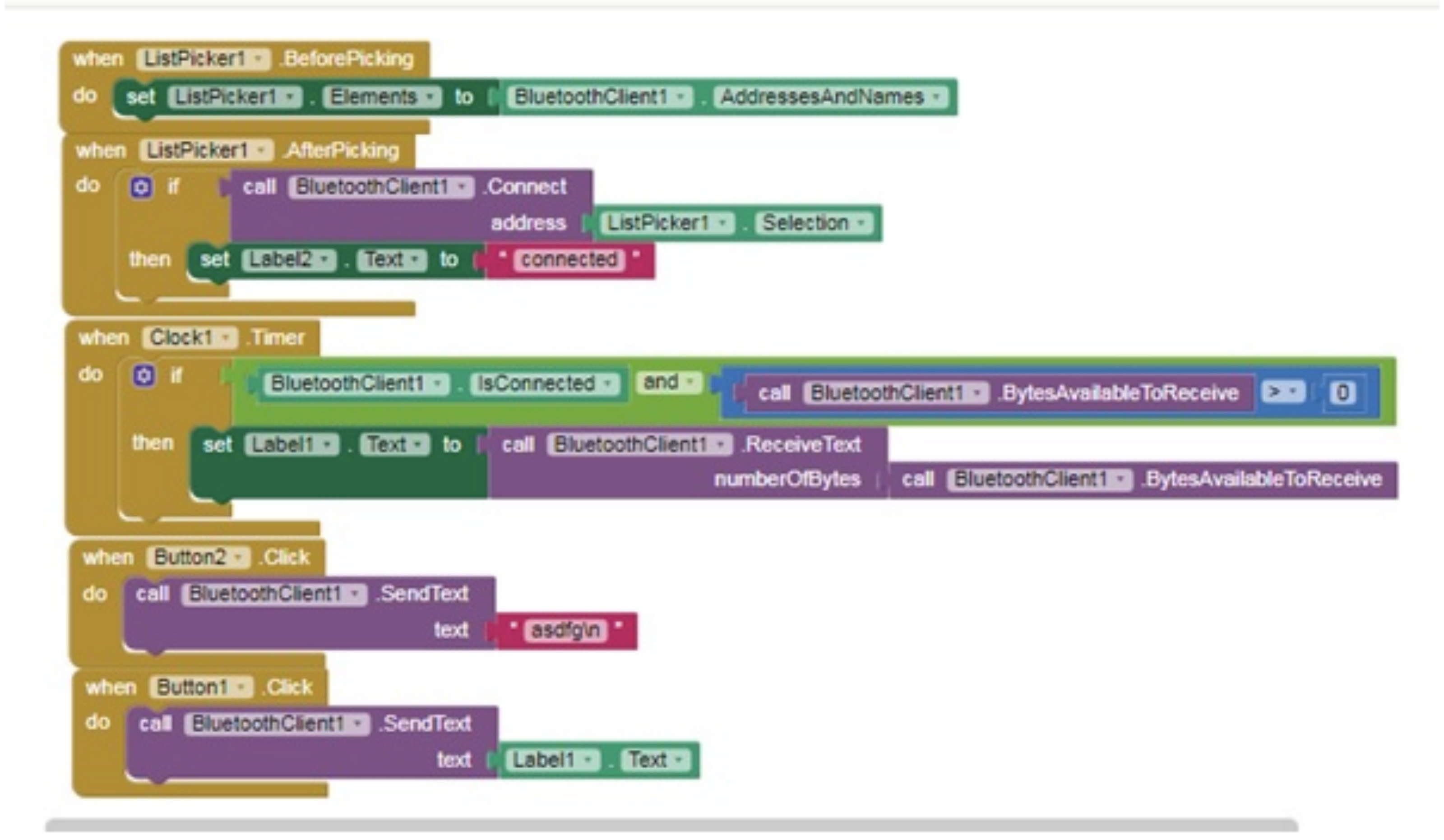
Now let's get into the code Blocks tab, for coding.

First, we need to initialise Bluetooth list available for connection. Then we have to set button function to send device id according to our Arduino Code as in the pic below (Fig. 6).

Now, export app .apk and install it on your Android phone.

You can get .apk, .aia file, and code from the link below.

Download Source code



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
when ListPicker1 .AfterPicking
do  if  call BluetoothClient1 .Connect
                          address ListPicker1 . Selection
    then set Label2 . Text to " connected "
when Clock1 .Timer
do  if  BluetoothClient1 . IsConnected and call BluetoothClient1 .BytesAvailableToReceive > 0
    then set Label1 . Text to call BluetoothClient1 .ReceiveText
                                  numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
when Button2 .Click
do  call BluetoothClient1 .SendText
                          text " asdfg\n "
when Button1 .Click
do  call BluetoothClient1 .SendText
                          text Label1 . Text
```

Fig. 6 MIT app inventor coding blocks

We are almost done now, we have completed our coding and app building. Let's set up the components and begin connections.

Connecting Components:

Arduino Pins	**Components and Pin**
Arduino Pin 9(pwm)	Servo Yellow Wire (signal input wire)
Rx	Bluetooth Module TX
Tx	Bluetooth Module RX
GND	GND Bluetooth
VCC	Bluetooth VCC
Pin 12	LED
GND	Battery(-VE)

Whole circuit and connections are illustrated in the pic below (Fig. 7).

We have connected all the components now. You might not like to fry costly Arduino board with the wrong connection. So, always crosscheck your setup to ensure that all connections are ok.

Hurray! We have now completed our awesome project on OTP-based lock system. Just plug it and follow the steps in the video for testing

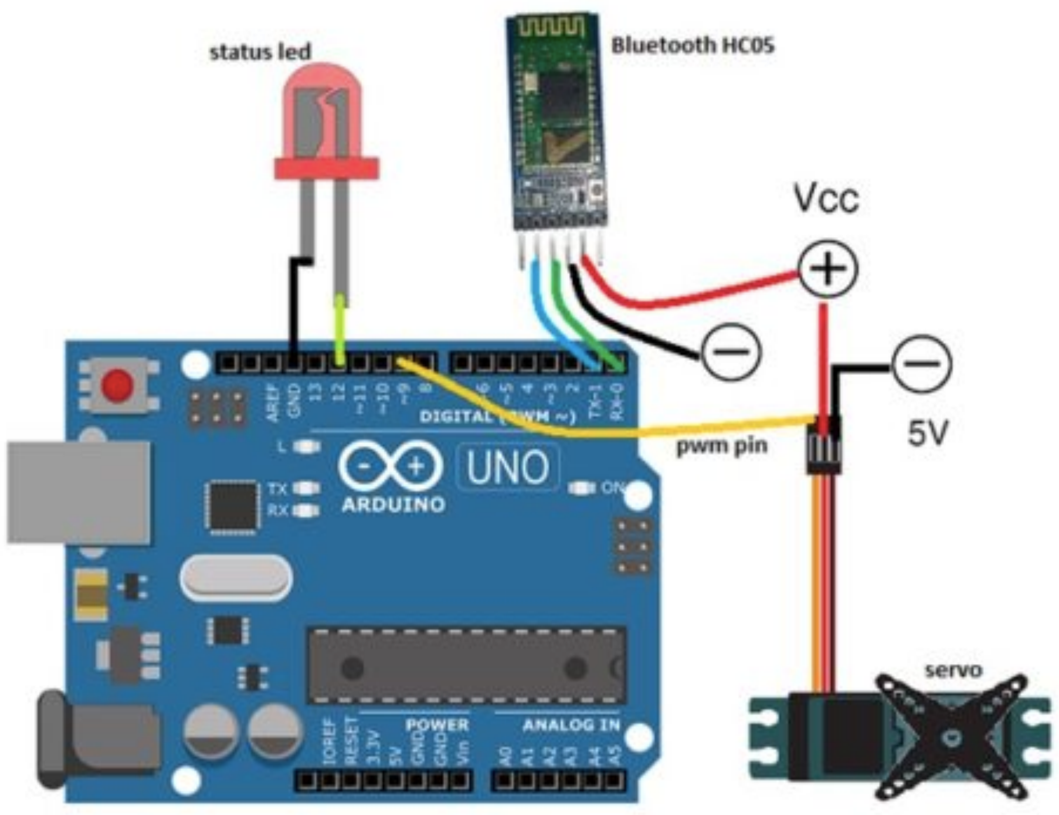


Fig. 7 Connections of components.

Testing:

First, connect the Arduino and components to a power supply. Here I have used 5V Power Bank for it. Next, open the installed App, then turn on the Bluetooth of the phone. When you tap on the Bluetooth icon, you will get the list of Bluetooth connections for pairing. Now, tap on HC 05. On successful pairing, you will get a ‘connected’ message on the App as in Fig 8.

After that, tap on the key icon to send device id to match. If the device id is matched, it will send an OTP to your app that you can see in App text bar between Bluetooth and lock icon.

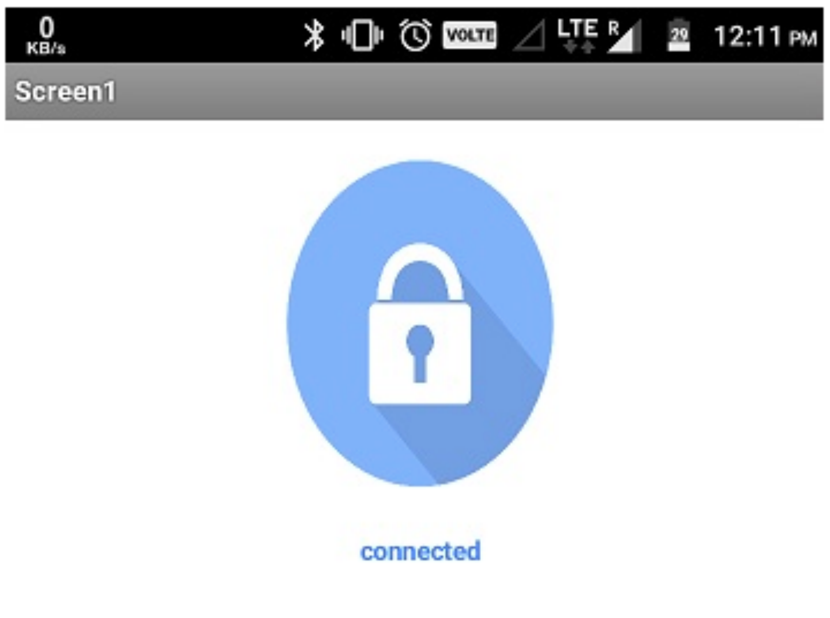


Fig. 8 Showing connection status

Now, you can tap on the lock icon to unlock your Smart Lock. If everything is ok then servo moves unlocking mechanism and onboard LED of Arduino lights up indicating successful unlock. I have used servo because of its high torque, and also because we can control the angle of its movement that helps in unlocking mechanism ofthe lock.

FAQ:

Can I supply power to Servo from Arduino?

No, never do that. You might have seen in different sites, where the servo is directly connected to 5V pin of Arduino, but it not safe. Arduino is not made for that amount of current. Don’t try to fry your costly board.

You can watch Video for this DIY here: https://youtu.be/ZKc506OxQz0

10

HUMIDITY AND TEMPERATURE MONITORING USING ARDUINO WITH THE IOT

BY D. CITHARTHAN AND T. JONAH GILBERT

Using the Internet of Things (IoT) in homes and industries it is possible to control any electrical or electronic equipment. Moreover, you can get the information from any sensor and analyse it graphically or in any user-defined format from anywhere in the world. The IoT using Arduino microcontroller (MCU) is easy and fun for those who are new to the field. Presented here is a humidity and temperature monitoring using Arduino.

In this article, humidity and temperature information from DHT-11 sensor is analysed graphically on ThingSpeak platform using Arduino MCU and ESP8266 Wi-Fi module. Block diagram of the whole setup is shown in Fig. 1.

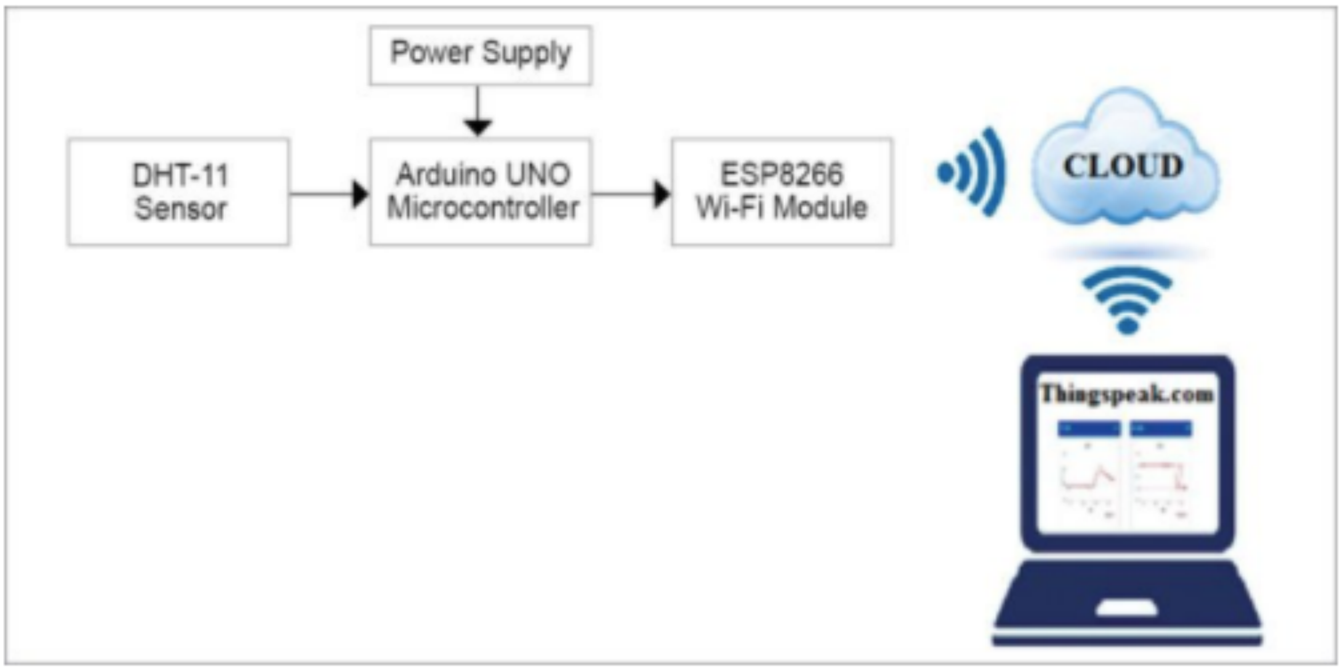


Fig. 1: Block diagram of humidity and temperature monitoring using Arduino with ESP8266

Circuit and working

Circuit diagram for monitoring humidity and temperature is shown in Fig. 2. It is built around Arduino MCU, DHT11 sensor and ESP8266 Wi-Fi module.

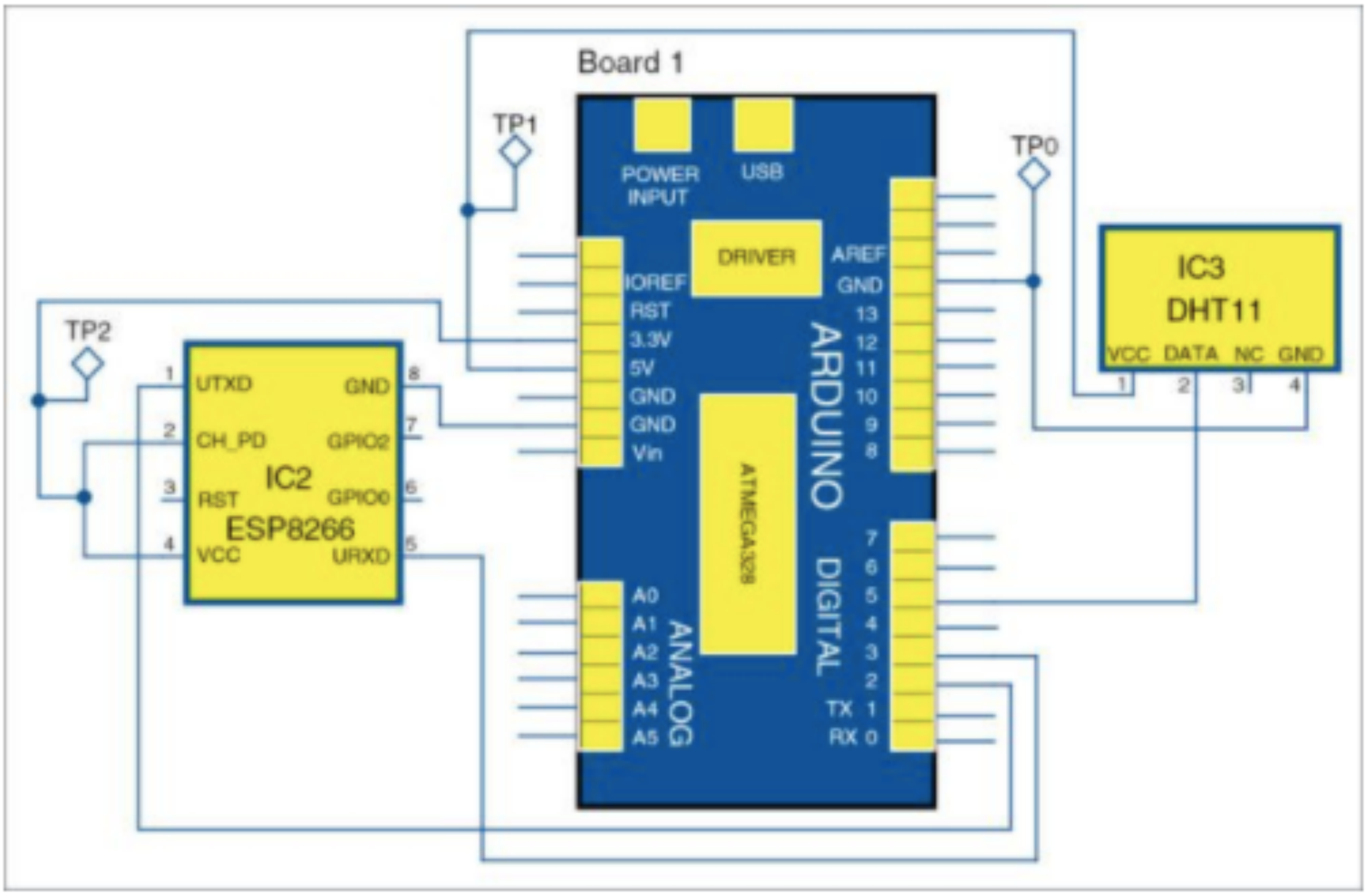
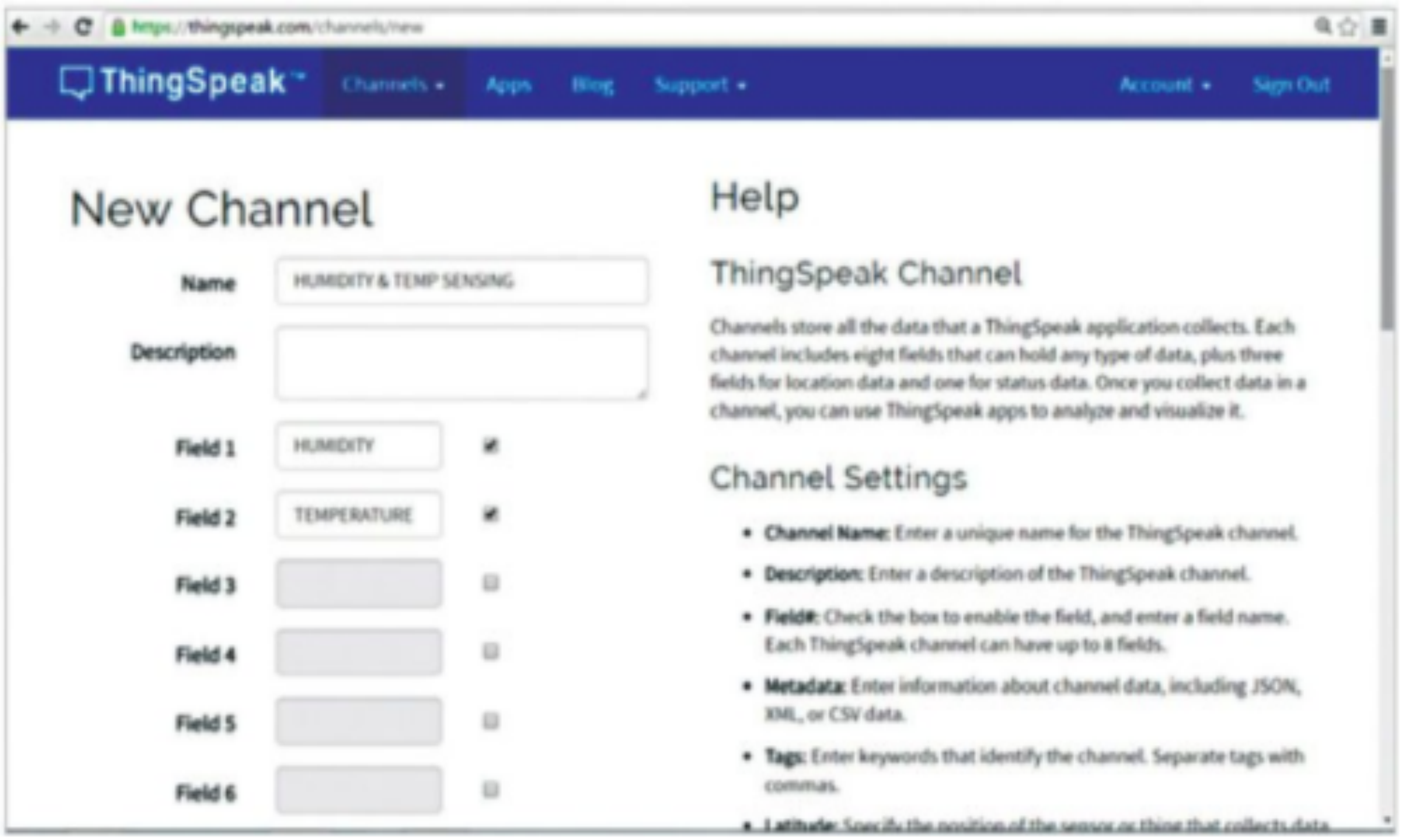


Fig. 2: Circuit diagram humidity and temperature monitoring using Arduino with ESP8266

The DHT11 sensor senses humidity and temperature, and sends the information to digital pin 5 of Arduino MCU, as shown in Fig. 2. From Arduino MCU, humidity and temperature values are uploaded to the Cloud at regular intervals of time through ESP8266 Wi-Fi module. From the Cloud, humidity and temperature values can be seen graphically on ThingSpeak platform from anywhere in the world.

Test Points

Test point	Details
TP0	0V (GND)
TP1	5V
TP2	3.3V



New Channel

Name	HUMIDITY & TEMP SENSING	
Description		
Field 1	HUMIDITY	✓
Field 2	TEMPERATURE	✓
Field 3		☐
Field 4		☐
Field 5		☐
Field 6		☐

Help

ThingSpeak Channel

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Latitude:** Specify the position of the sensor or thing that collects data.

Fig. 3: New channel on ThingSpeak platform

Construction and testing

ThingSpeak is an open source data platform but you need to register to use it. After registering, login to your account and create a new channel with humidity as one field and temperature as another, as shown in Fig. 3. Once a new channel is created, it will generate two API keys, namely, write API key and read API key. Replace the line given below in the program with your write API key:

String apiKey = " NTIM1RXET6YVUVWF ";

Next, substitute Host_Name and Password with your Wi-Fi name and Wi-Fi password in the two lines given below in the program (IoT.ino):

String Host_Name = "Jonah";
String Password = "2569696";

The program should be verified with your Wi-Fi setup. It uses DHT library. If DHT library is not present in your Arduino folder, download it from https://github.com/adafruit/DHT-sensor-library. To import DHT library in Arduino IDE, select Sketch→Import library→Add library→Select the library that you have downloaded.

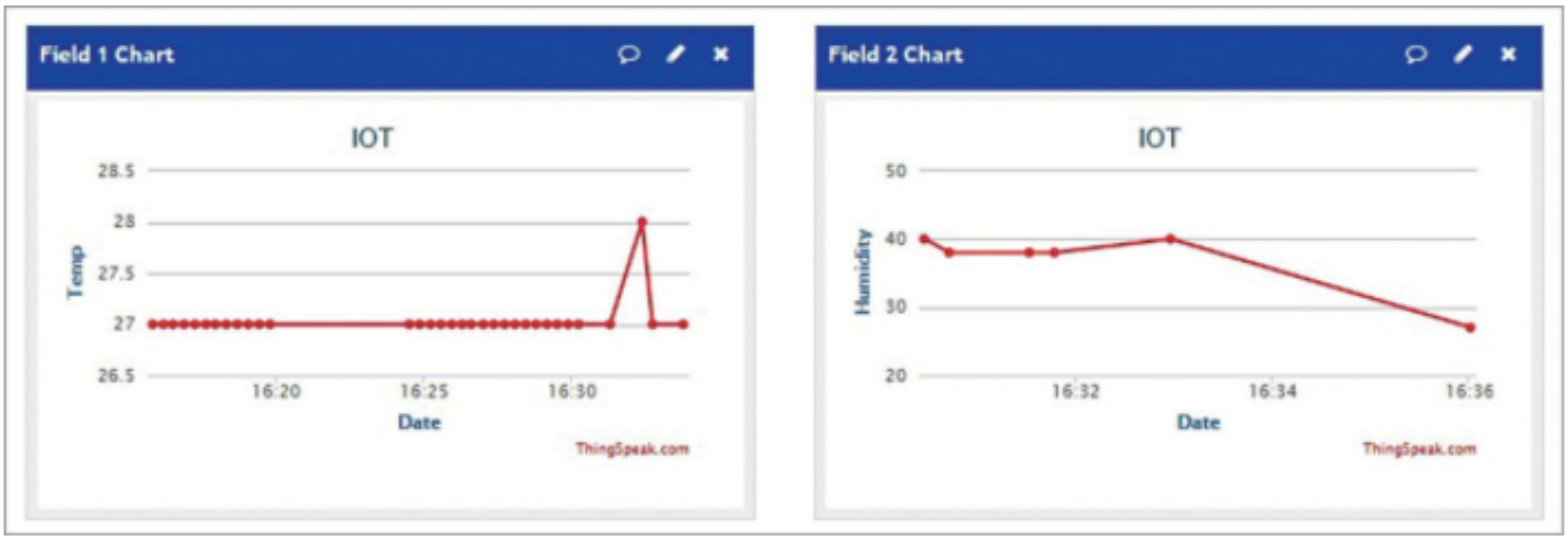
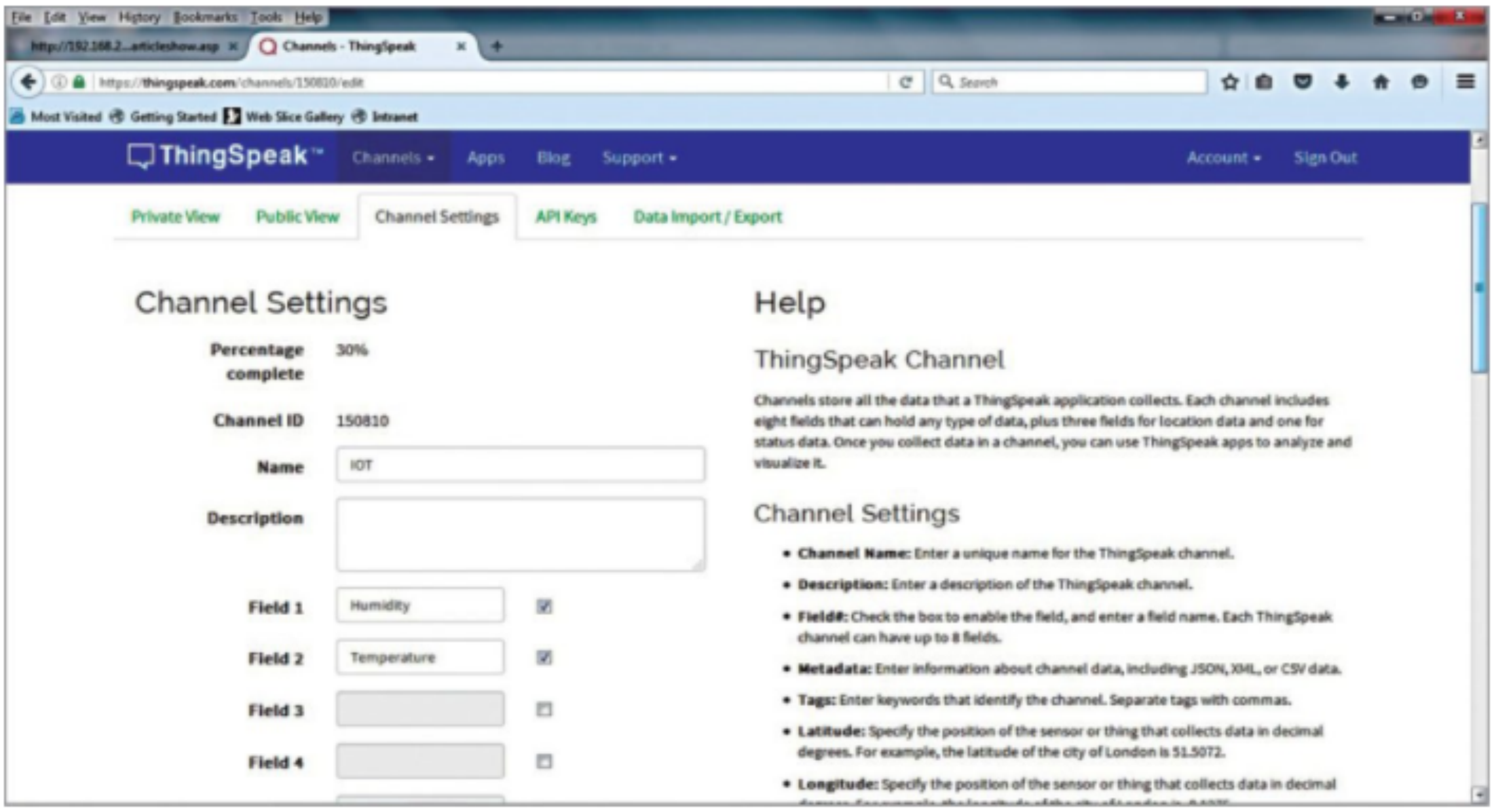


Fig. 4: Graphical view of humidity and temperature on ThingSpeak platform

Compile the sketch/program and upload it to Arduino MCU through Arduino IDE. Ensure that Wi-Fi modem and the Internet connection in your PC/smartphone are working properly.

Once sketch uploading is done, it will upload humidity and temperature values on ThingSpeak platform and you will be able to see it graphically in Private View



Channel Settings

Percentage complete	30%
Channel ID	150810
Name	IOT
Description	
Field 1	Humidity ✓
Field 2	Temperature ✓
Field 3	☐
Field 4	☐

Help

ThingSpeak Channel

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Latitude:** Specify the position of the sensor or thing that collects data in decimal degrees. For example, the latitude of the city of London is 51.5072.
- **Longitude:** Specify the position of the sensor or thing that collects data in decimal

Fig. 5: Channel Settings

window, as shown in Fig. 4. If you want to change channel or field name, you can change it from Channel Settings (Fig. 5). Author's prototype is shown in Fig. 6.

Other applications

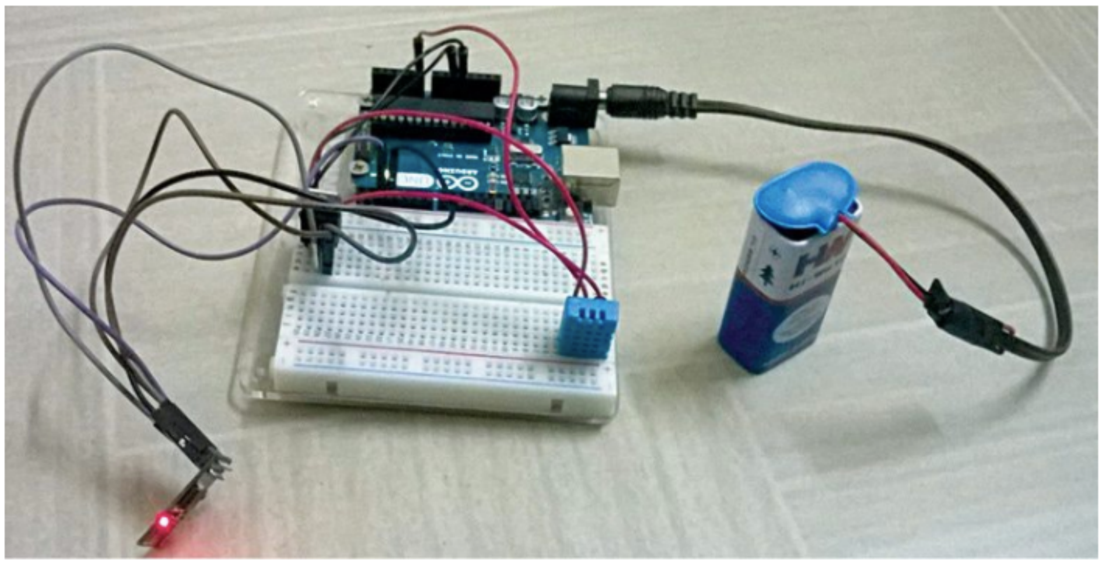
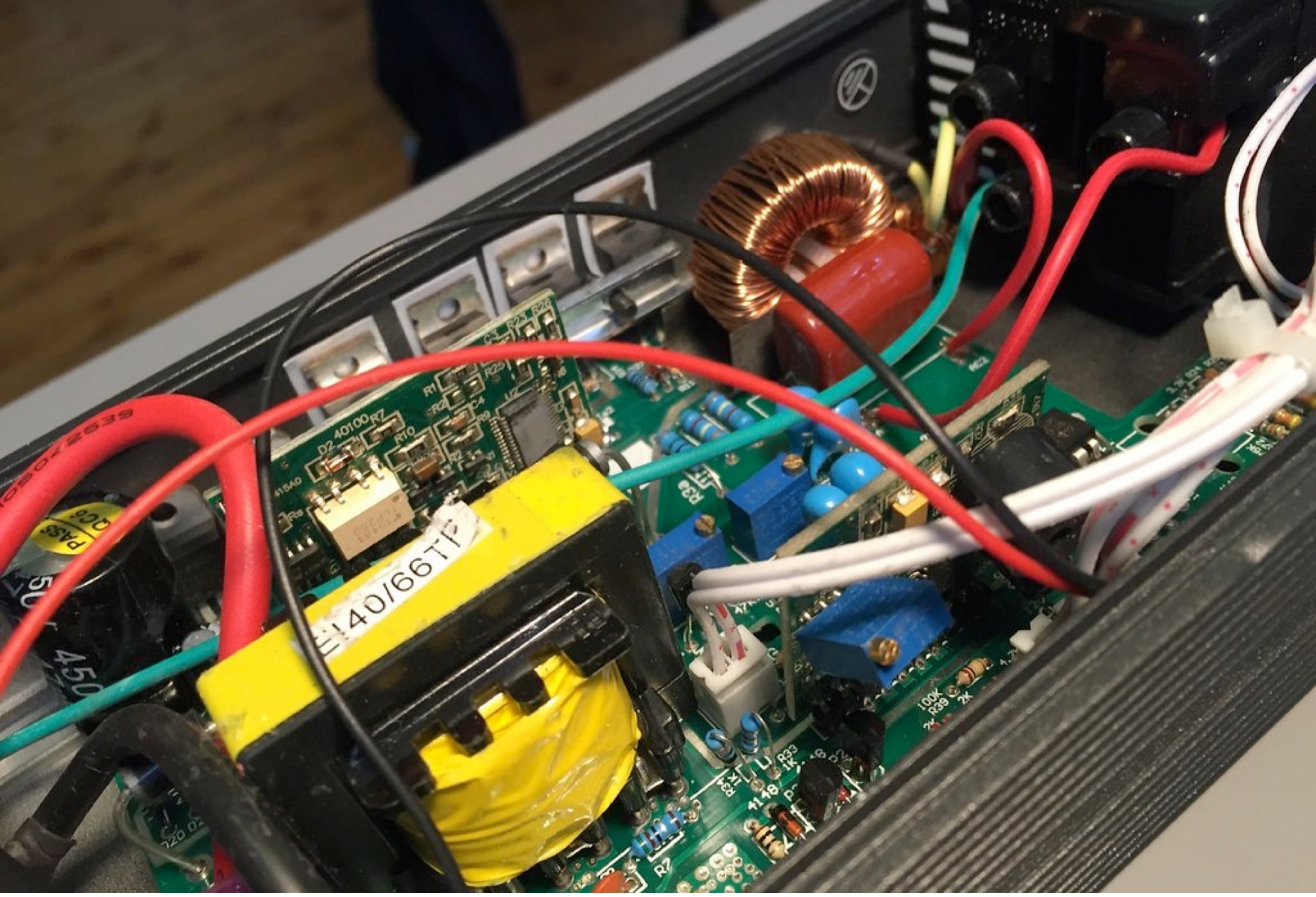


Fig. 6: Author's prototype of the humidity and temperature monitoring using Arduino with ESP8266

Along with temperature and humidity sensor, other sensors like gas, voltage, current and energy can be used based on the requirement. Moreover, it is also possible to take further actions by controlling the actuators from the Internet once the sensor values are going above/below predetermined values.

Download Source folder



11

1KW SINE WAVE INVERTER

DR R.V. DHEKALE

An inverter provides power backup for mains-based appliances in the event of a power failure. Most of the inverters available in the market have complicated circuit design and are not very economical. Some of them produce a square-wave output, which is undesirable for inductive loads. The project is a simple sine wave inverter circuit that produces 50Hz quasi-sine wave output using a single IC CD4047 and some discrete components, which makes it a very cost-effective solution.

Sine wave inverter circuit description

Fig. 1 shows the sine wave inverter circuit of the MOSFET-based 50Hz inverter. It comprises a CD4047 multivibrator (IC1), IRF250 MOSFETs (T1 through T8), transistors and a few discrete components.

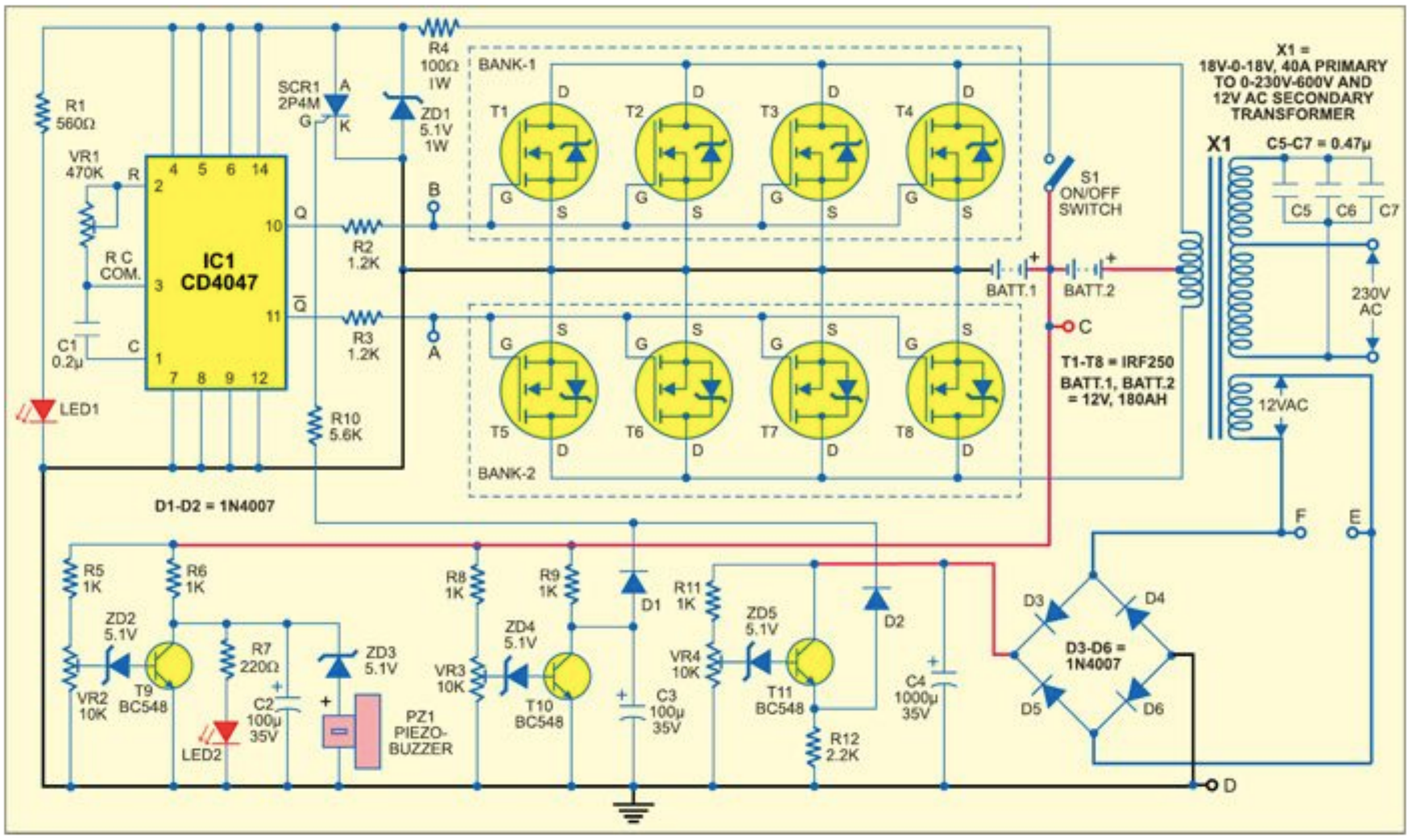
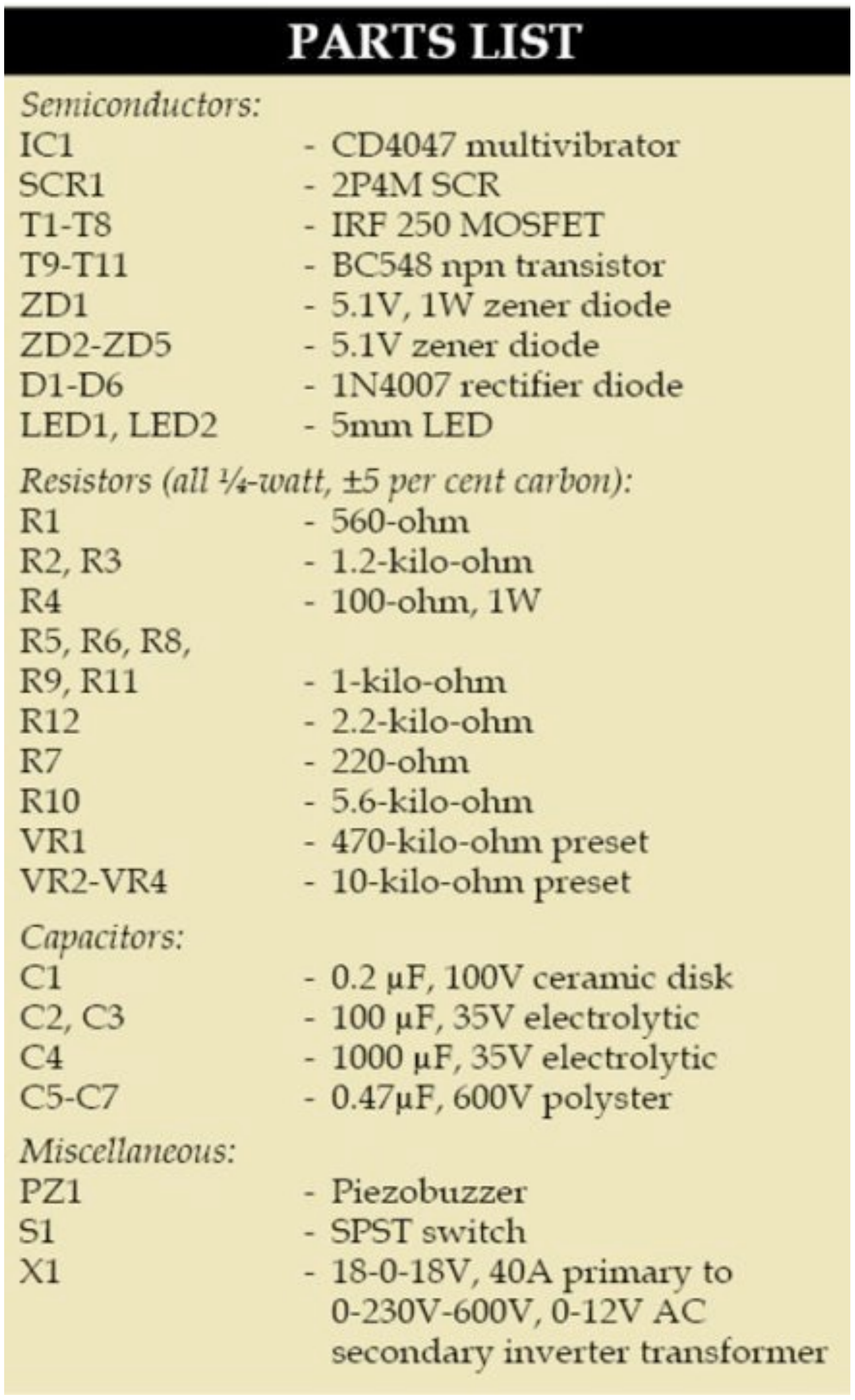


Fig. 1: Sine wave inverter circuit

IC CD4047 has built-in facilities for astable and bistable multivibrators. The inverter application requires two outputs that are 180 degrees out of phase. Therefore IC1 is wired to produce two square-wave output signals at pins 10 and 11 with 50Hz frequency, 50 per cent duty cycle and 180-degree phase-shift. The oscillating frequency is decided by external preset VR1 and capacitor C1.

These two signals drive the two MOSFET banks (bank-1 and bank-2) alternatively. When pin 10 of IC1 is high and pin 11 low, MOSFETs of bank-1 (T1 through T4) conduct, while MOSFETs of bank-2 (T5 through T8) remain in the non-conducting state. Therefore a large swing of current flows through the first half of the primary winding of inverter transformer X1 and 230V AC develops across the secondary winding.

During the next half cycle, the voltage at pin 10 of IC1 goes low, while the voltage at pin 11 is high. Thus MOSFETs of bank-2 conduct, while the MOSFETs of bank-1 remain non-conducting. Therefore current flows through the other half of the primary winding and 230V AC develops across the secondary winding.



PARTS LIST

Semiconductors:	
IC1	- CD4047 multivibrator
SCR1	- 2P4M SCR
T1-T8	- IRF 250 MOSFET
T9-T11	- BC548 npn transistor
ZD1	- 5.1V, 1W zener diode
ZD2-ZD5	- 5.1V zener diode
D1-D6	- 1N4007 rectifier diode
LED1, LED2	- 5mm LED
Resistors (all ¼-watt, ±5 per cent carbon):	
R1	- 560-ohm
R2, R3	- 1.2-kilo-ohm
R4	- 100-ohm, 1W
R5, R6, R8, R9, R11	- 1-kilo-ohm
R12	- 2.2-kilo-ohm
R7	- 220-ohm
R10	- 5.6-kilo-ohm
VR1	- 470-kilo-ohm preset
VR2-VR4	- 10-kilo-ohm preset
Capacitors:	
C1	- 0.2 µF, 100V ceramic disk
C2, C3	- 100 µF, 35V electrolytic
C4	- 1000 µF, 35V electrolytic
C5-C7	- 0.47µF, 600V polyster
Miscellaneous:	
PZ1	- Piezobuzzer
S1	- SPST switch
X1	- 18-0-18V, 40A primary to 0-230V-600V, 0-12V AC secondary inverter transformer

This way an alternating output voltage is obtained across the secondary winding.

The sine wave output is obtained by forming a tank circuit with the secondary winding of the inverter transformer in parallel with capacitors C5 through C7. Two 2.2µF capacitors are connected to the gates of the MOSFETs in both the banks with respect to the ground if proper sinewave is not produced. Natural frequency of the tank circuit is adjusted to 50 Hz. Current consumption with no load is only 500 mA due to 50 per cent duty cycle of the square-wave signal. As the load is increased, current consumption increases.

The supply voltage to IC1 is limited to 5.1 volts by using zener ZD1 and resistor R4 with the external battery as shown in Fig. 1.

Low-battery indicator

The low-battery indication circuit consists of transistor T9, preset VR2, zener diode ZD2, resistors R5, R6 and R7, LED2 and capacitor C2. The 12V supply voltage from BATT.1 is applied to the low-battery indicator circuit with full load (not more than 1000 watts) connected to the inverter output. The voltage across the load is 230V AC. At this instant, adjust preset VR2 such that zener diode ZD2 and transistor T9 conduct to drop the collector voltage to 0.7 volt keeping LED2 ‘off.’

If supply voltage goes below 10.5 volts, the voltage across the load decreases from 230V AC to 210V AC. At this instant, zener diode ZD2 and transistor T9 do not conduct and hence the collector voltage increases to about 10.5 volts and LED2 glows to indicate low voltage of the battery. At the same time, piezobuzzer PZ1 produces an audio tone indicating low battery.

Low-battery cut-off

If the battery is discharged to zero volt repeatedly, the battery life will decrease. The low-battery cut-off circuit consists of transistor T10, preset VR3, zener diode ZD4, resistors R8 and R9, capacitor C3 and diode D1.

Adjust preset VR3 such that when the voltage across the load is above 200 volts, zener diode ZD4 and transistor T10 conduct. The collector voltage of T10 is about 0.7 volt in this case and hence the SCR (SCR1) will not conduct.

But if the voltage across the load goes below 200 volts, zener diode ZD4 and transistor T10 will not conduct and the collector voltage of T10 will increase, causing the SCR to conduct.

Once the SCR conducts, the supply voltage to IC1 (CD4047) will be 0.7 volt, due to which IC1 will be unable to produce the voltage pulses at output pins 10 and 11 and the inverter will turn off automatically. During this state, the SCR remains conducting.

Low cut-off of the inverter can be set at the load voltage of 170 volts for tubelight, fan, etc. So the tubelight and fan will not be switched off until the voltage goes below 170 volts.

No-load cut-off

If there is no load connected at the output of the inverter, the output voltage is 270 to 290 volts. This voltage is sensed by the 0-12V tap at the secondary winding of inverter transformer X1, which is connected to the no-load cut-off circuit comprising zener diode ZD5, transistor T11, preset VR4, resistors R12 and R11, and capacitor C4.

When no load is connected, the voltage at 12V tap will also increase. This voltage is rectified by the full-wave bridge rectifier comprising diodes D3 through D6, filtered by capacitor C4 and given to transistor T11.

Adjust preset VR4 such that if the inverter voltage goes above 250 volts, zener diode ZD5 and transistor T11 conduct. This increases the emitter voltage, hence the SCR fires to switch the inverter 'off.' When proper load is connected, the inverter will automatically turn on.

Construction

An actual-size, single-side PCB for the sine wave inverter circuit is shown in Fig. 2 and its component layout in Fig. 3. Suitable connector CON1 is provided on the PCB to connect the MOSFET banks and the transformer externally. Connector CON1 pins A through F are also marked on schematic. Assemble the circuit on a PCB as it saves time and minimises assembly errors. Carefully assemble the components and double-check for any overlooked error. MOSFETs should be mounted over heat-sinks using mica spacers as the insulators between them.

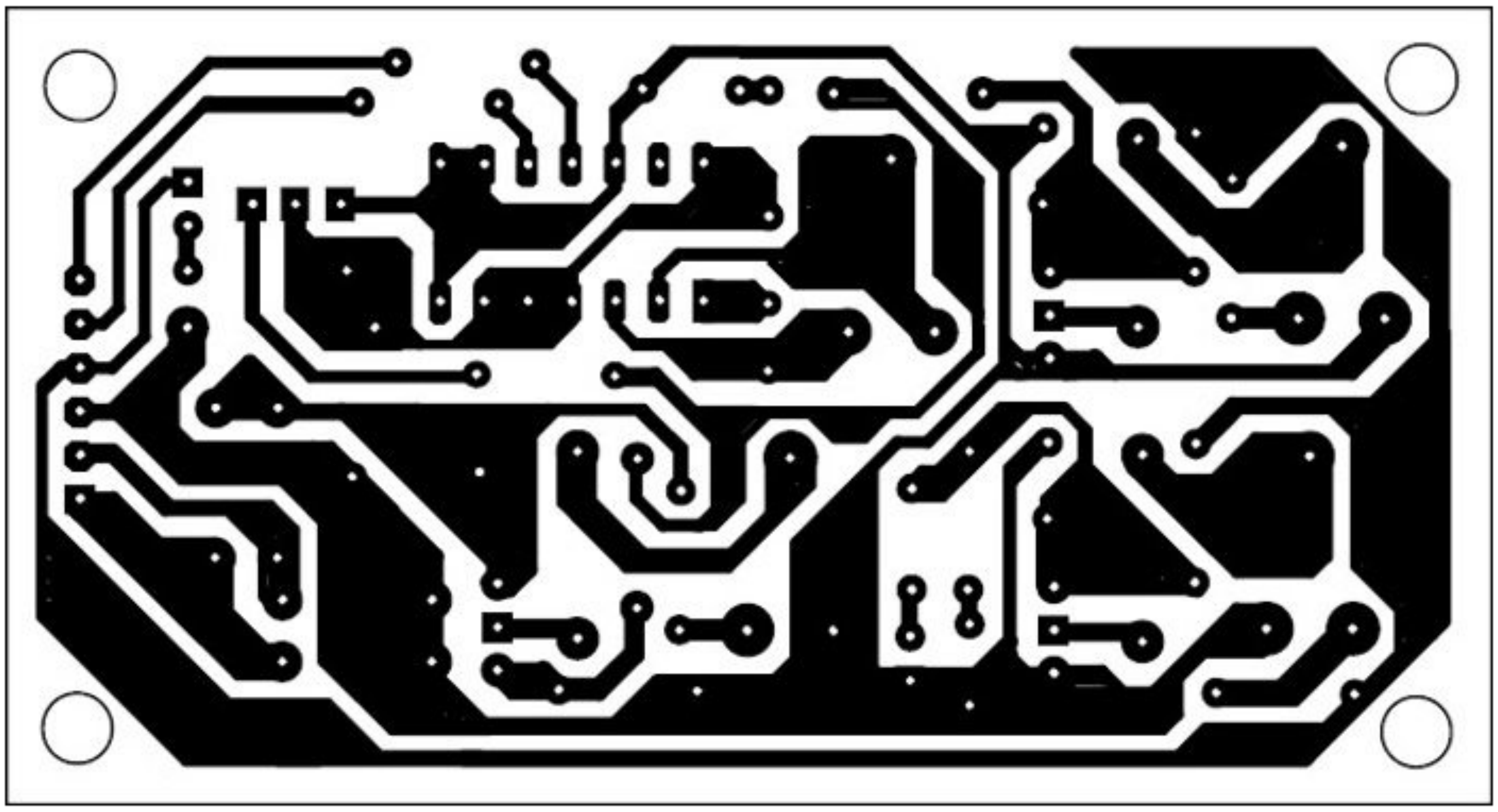


Fig. 2: An actual-size, single-side PCB for the sine wave inverter circuit

Connect the 24V supply terminal directly to the centre tap of the primary winding of the inverter transformer, which carries maximum current of more than 50 amperes with 1000 watts. Current depends on the load applied. There is no need to add a switch in the high-current path to make the inverter turn on and off. The inverter can be switched on and off by low-current switch S1.

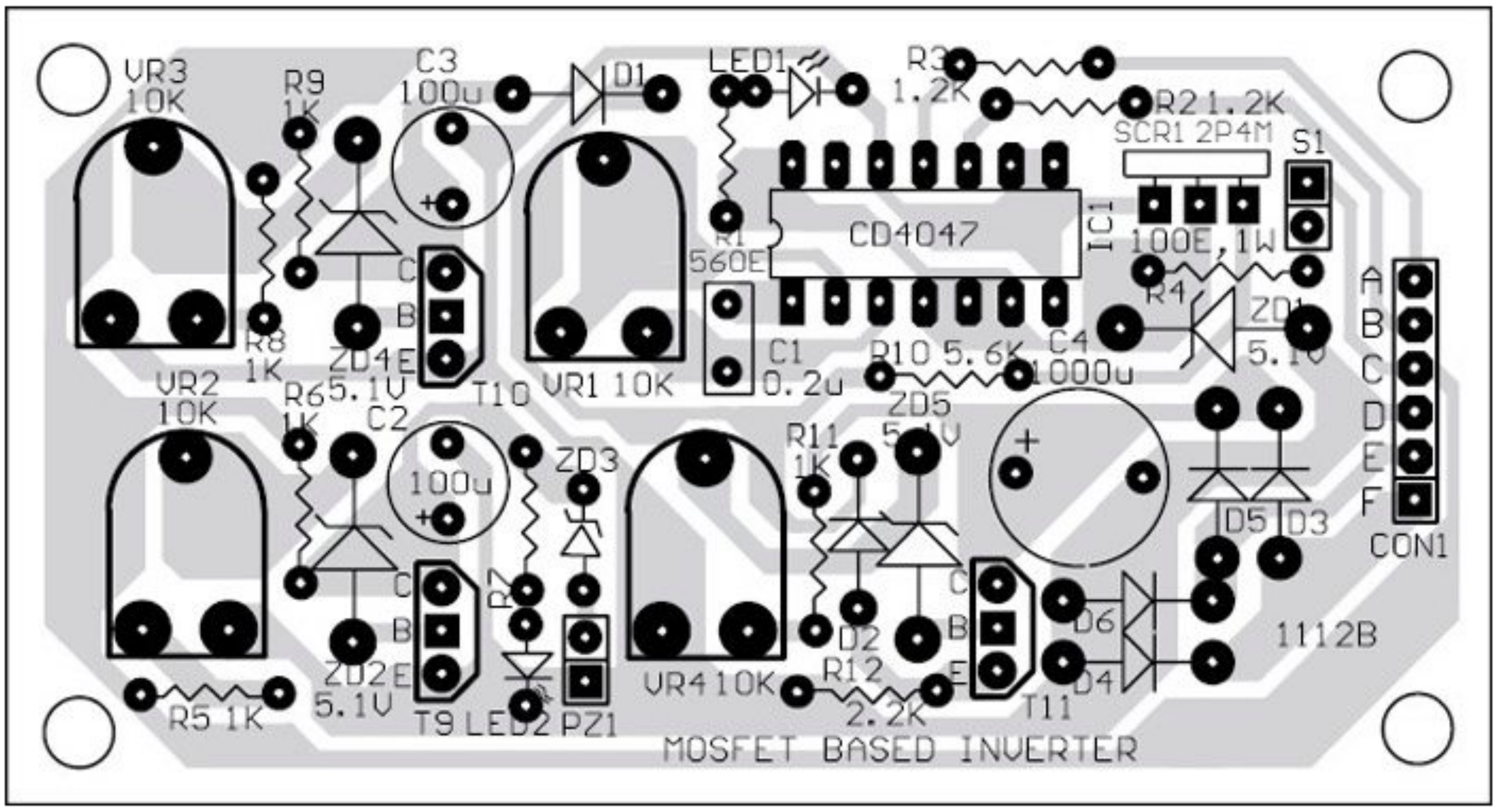


Fig. 3: Component layout for the PCB

Download PCB and Component Layout PDFs: click here

12

NOISE DETECTOR FOR SILENCE ZONE AREA

BY ASHWINI KUMAR SINHA

Please Keep Silence! You must have seen this message written in places like library and office. But some people intentionally or unintentionally tend to make noise in such silence zone areas, disturbing the others. How about having a device that can help us maintain silence in such silence zone areas?

Here, in this project we are going to make a very sensitive smart noise detector that will identify loud talker and alert us by displaying red light and trigging buzzer.

We can also connect this device to our phone to set its sensitivity level according to our choice.

Let's start the project.

First, we need to have these components -

Components

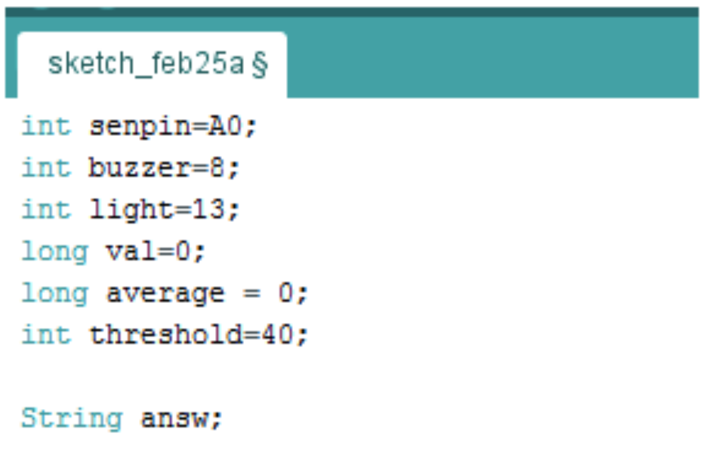
- Arduino Uno
- Red LED bulb
- Buzzer
- Sound Sensor.
- Some Jumper wires
- Battery/ 5v Power bank
- Bluetooth HC 05

Bill of materials

Component Name	Quantity	Description	Cost Approx. In INR
Arduino Uno	1	For Programing	300
Sound Sensor Module	1	FOR USB HID	100
Bluetooth HC 05	1	300	300
Wires	20cm	For Connection	10
Buzzer	2	For mouse click	50
Battery	1	Power Source	30
Red Led	1	For Indication	2
Total Cost			792

Coding

After collecting the components, we will set up the variables for storing values. Refer (Fig .1).

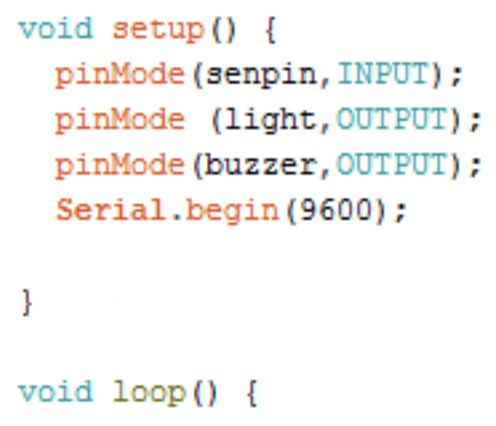

sketch_feb25a §

```
int senpin=A0;
int buzzer=8;
int light=13;
long val=0;
long average = 0;
int threshold=40;

String answ;
```

Fig .1 Setting variables

Then we will set up Bluetooth and I/O pin for input and output in setup function.

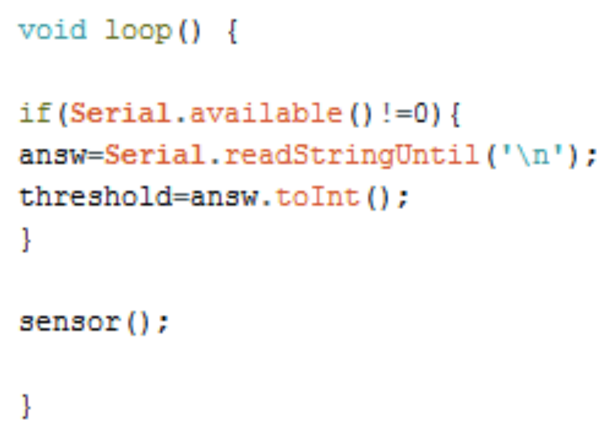
Here, I have set up the Bluetooth baud rate with 9600 but if didn't work you can try default baud rate 38400.



```
void setup() {
  pinMode(senpin,INPUT);
  pinMode (light,OUTPUT);
  pinMode(buzzer,OUTPUT);
  Serial.begin(9600);

}

void loop() {
```

Fig 2. Arduino code setting I/O pins

After this, we have to set loop functions for running code repeatedly (Refer to Fig .3) In this loop function, we create a if() condition that listens to Bluetooth incoming string, then we convert that string to integer value and then set that integer value to threshold value.



```
void loop() {

if(Serial.available()!=0){
answ=Serial.readStringUntil('\n');
threshold=answ.toInt();
}

sensor();

}
```

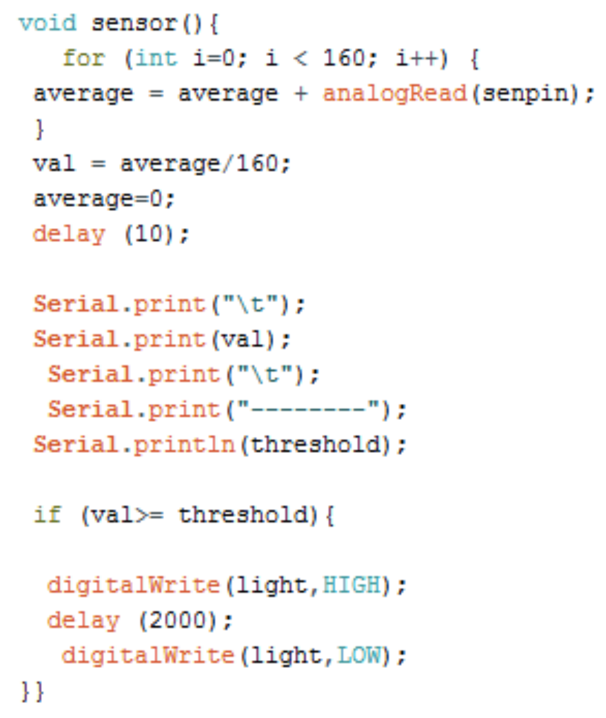
Fig .3 Setting loop function.

In the next part of coding, we create a sensor () function to get sensor value (refer Fig 4). For sensor value, we take an average of several readings for detecting noise. Formula for obtaining value of sensor is

Sum of all readings/ Total number of readings taken

Here we are going to take average of 160 readings. So, the formula is

Sum of all readings / 160.



```
void sensor(){
   for (int i=0; i < 160; i++) {
 average = average + analogRead(senpin);
 }
 val = average/160;
 average=0;
 delay (10);

 Serial.print("\t");
 Serial.print(val);
  Serial.print("\t");
  Serial.print("--------");
 Serial.println(threshold);

 if (val>= threshold){

  digitalWrite(light,HIGH);
  delay (2000);
   digitalWrite(light,LOW);
}}
```

Fig 4. Arduino code sensor function.

Building App

Create a layout as in Fig below and add the following components to it

- 2 Text level
- 1 Text input
- 1 List picker
- 1 Button

Now, go to MIT App inventor coding blocks

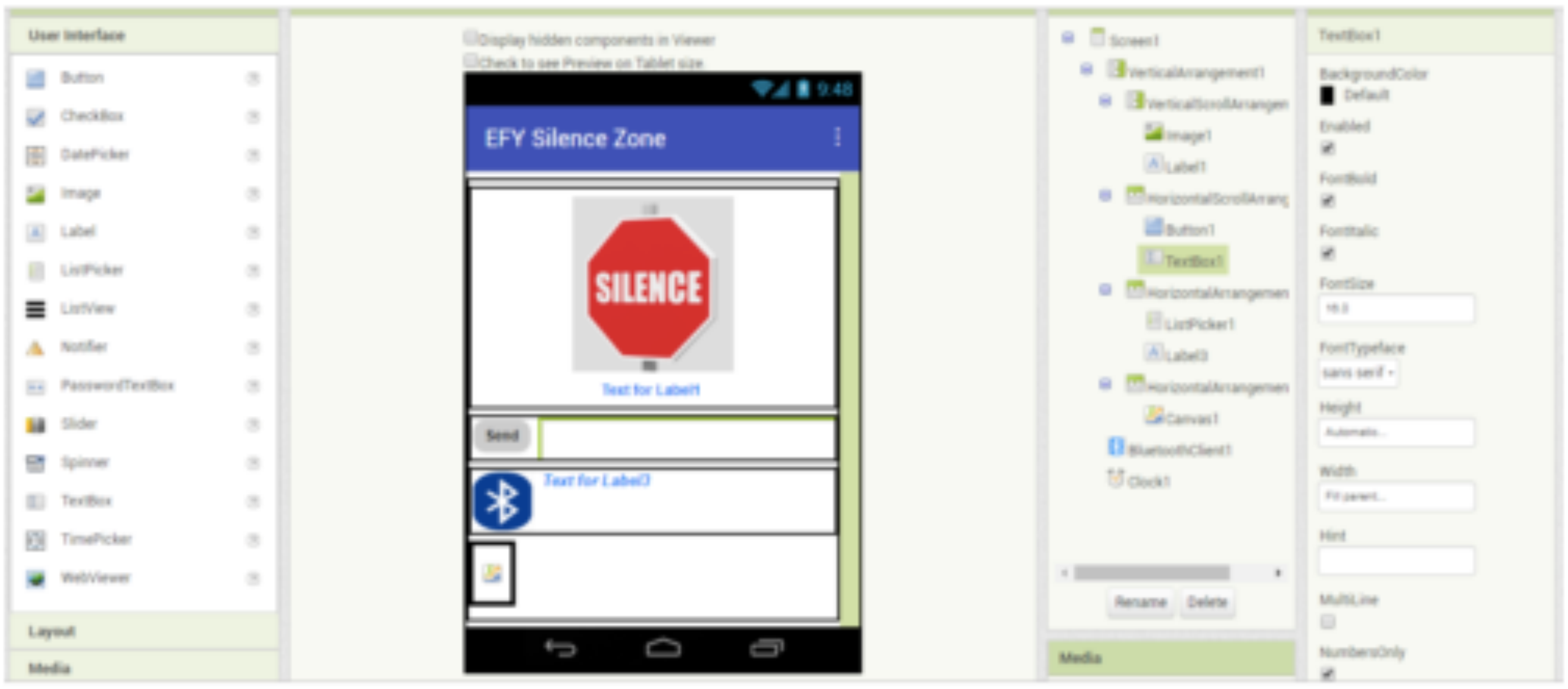
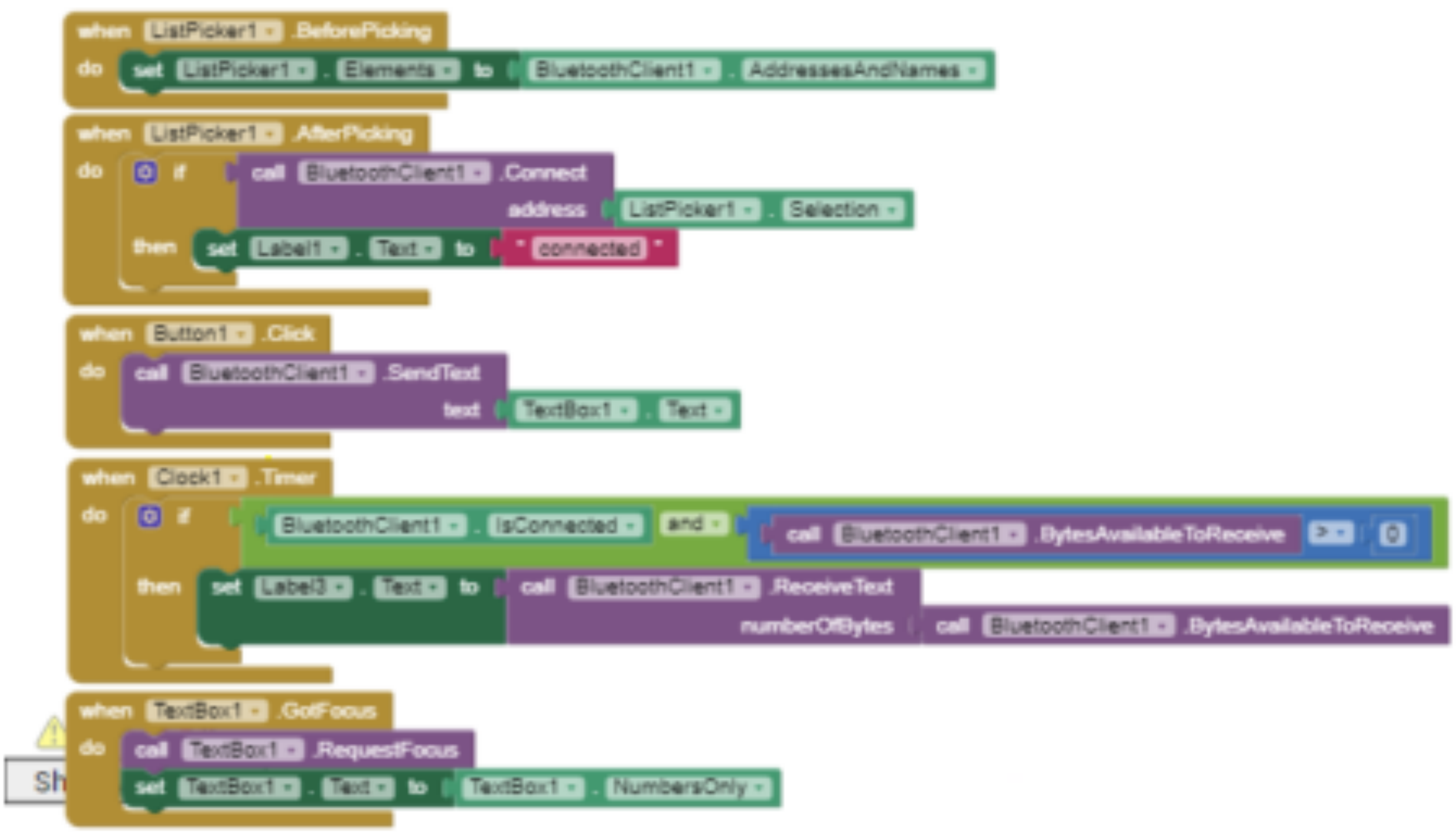


Fig 5. APP layout.



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  if  call BluetoothClient1 .Connect
                              address  ListPicker1 . Selection
    then  set Label1 . Text to " connected "

when Button1 .Click
do  call BluetoothClient1 .SendText
                          text  TextBox1 . Text

when Clock1 .Timer
do  if  BluetoothClient1 . IsConnected and  call BluetoothClient1 .BytesAvailableToReceive > 0
    then  set Label3 . Text to  call BluetoothClient1 .ReceiveText
                                     numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive

when TextBox1 .GotFocus
do  call TextBox1 .RequestFocus
    set TextBox1 . Text to TextBox1 . NumbersOnly
```

Fig 6. MIT APP inventor Code Blocks

and set coding blocks according to Fig 6. You can download the whole source code and app from the link below.

Download

Connection

Connect the components as shown in the following illustration (refer Fig 8)

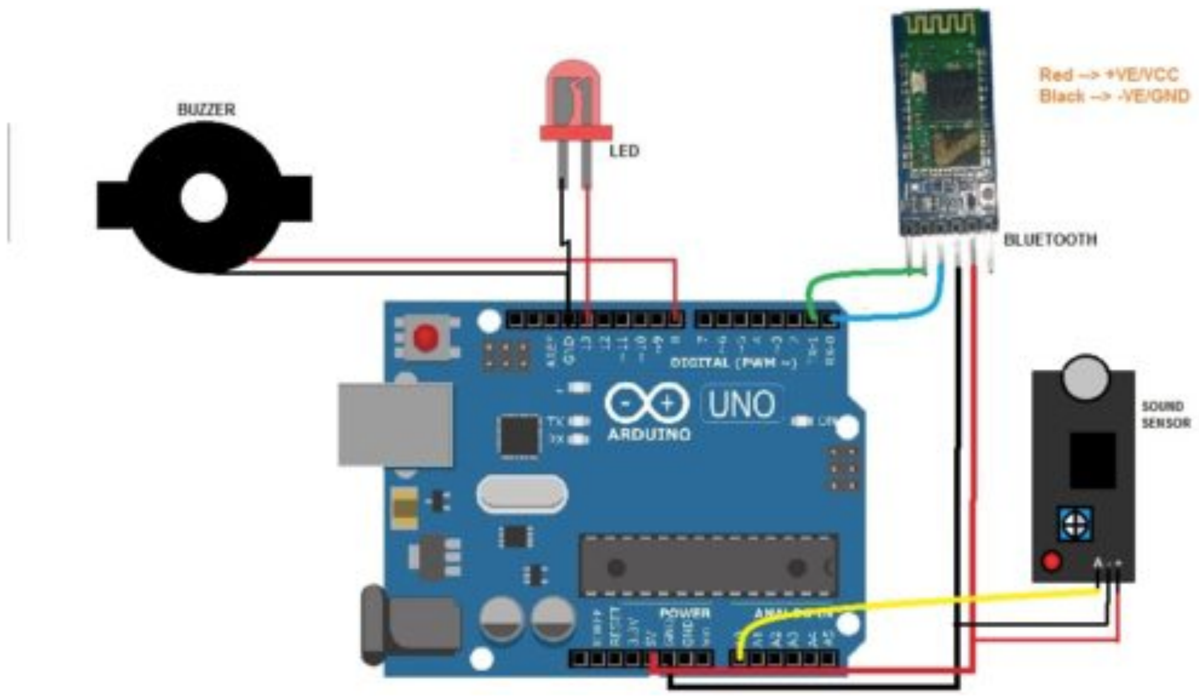


Fig .8 Connecting components

Arduino	Components
GND	BUZZER -VE
PIN 8	BUZZER +VE
TX	BLUETOOTH RX
RX	BLUETOOTH TX
5V	BLUETOOOTH VCC
GND	BLURTOOTH GND
PIN 13	LED +VE
GND	LED -VE
5V	SOUND SENSOR VCC
GND	SOUND SENSOR GND
ANALOG A0	SOUND SENSOR OUTPUT

Now our project is complete.

Testing

Power the Arduino and open the app tap at the Bluetooth Icon and select HC 05 from the list to connect your phone to Bluetooth. Now you can see the noise level value in the text bar. When noise level reaches the threshold value, the LED light and Buzzer will automatically get triggered by Arduino. You can also set threshold value using the app. To do so, tap on the text box next to 'send' button and enter the value you want to add and click on the 'send' button to set the threshold value.

Download Source Code

You can watch Video for this DIY here: https://youtu.be/zcgr5xlTz8w

13

SMS-BASED SMART NOTICE BOARD

BY ASHWINI KUMAR SINHA

We are all familiar with how notice boards in schools, offices, colleges and railway platforms look like. But have you ever wondered how they work and how you can make such a board by yourself?

So, today we are going to make an SMS-based notice board which can be updated just by a single message sent from your smartphone.

In order to begin, let's have a look at the required components.

Bill Of Material

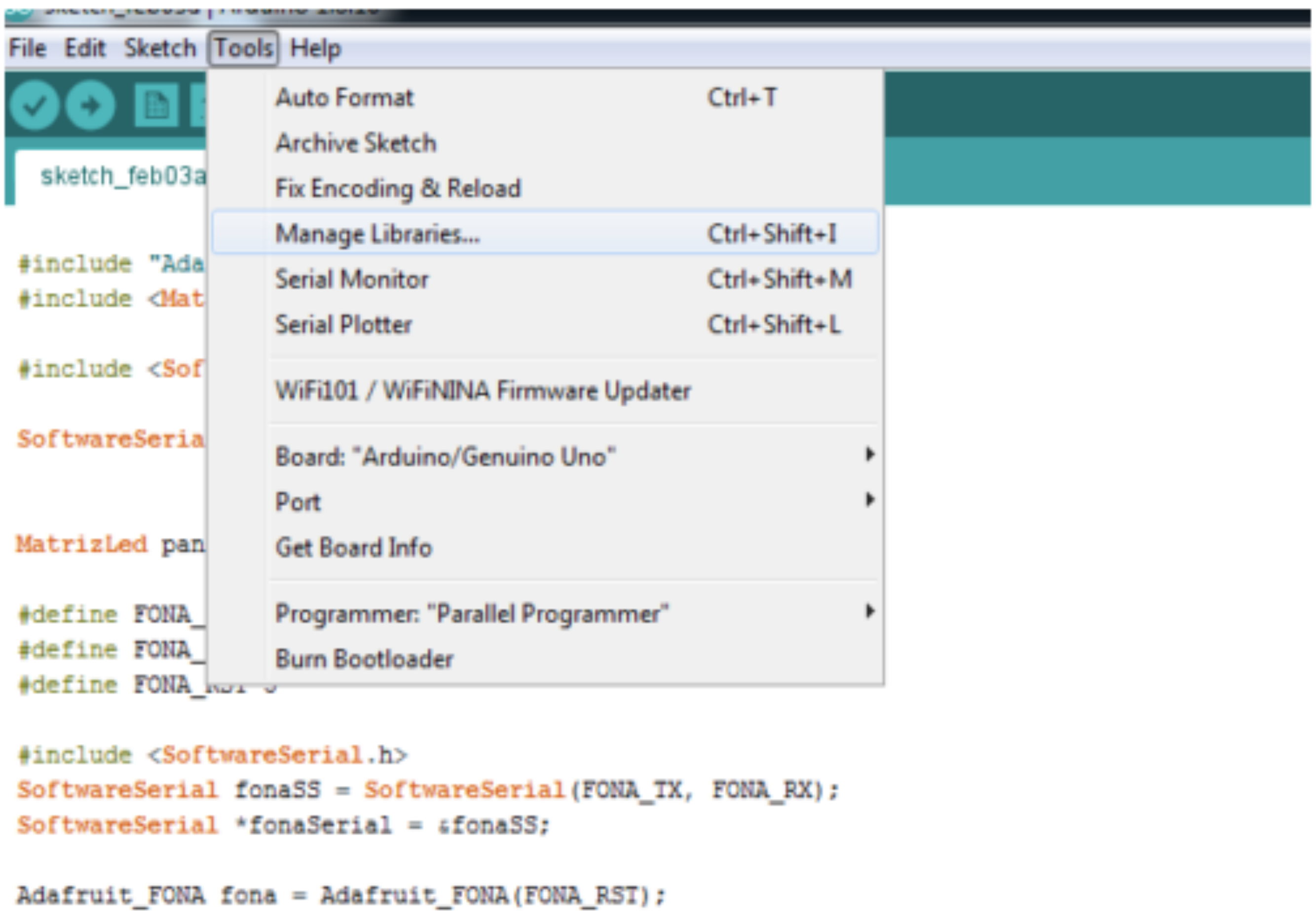
Component Name	Quantity	Description	Cost Approx. In INR
Arduino Uno/Nao/Mini	1	For Programming	200
MAX7219 Dot Matrix Display	1	Display	300
Sim800l	1	GSM Module	400
Wires	30cm	For Connection	30
Total Cost			**930**

Prerequisites

We have to install the required libraries in the Arduino IDE. To do so, follow the steps given below.

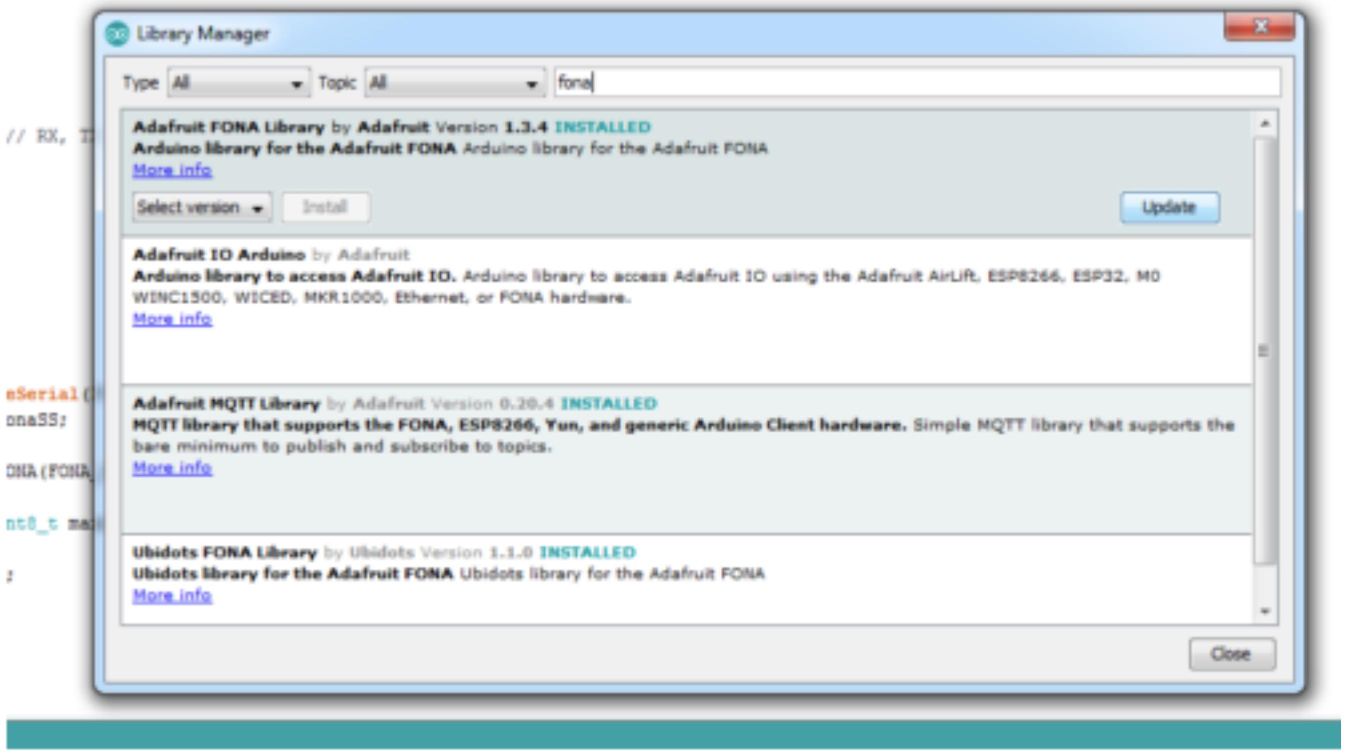
Click on Tools -> Manage library -> Search ”MatrizLed “ and click install.

Then install another library called “Adafurit_Fona”. (Refer Fig 2,3,4).


File Edit Sketch Tools Help
sketch_feb03a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "Parallel Programmer"
Burn Bootloader

```
#include "Ada
#include <Mat

#include <Sof

SoftwareSeria


MatrizLed pan

#define FONA_
#define FONA_
#define FONA_

#include <SoftwareSerial.h>
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

Adafruit_FONA fona = Adafruit_FONA(FONA_RST);
```

Fig 2.



```
// RX, T

eSerial(
onaSS;

ONA(FONA

nt8_t ma

;
```

Library Manager

Type All Topic All fona

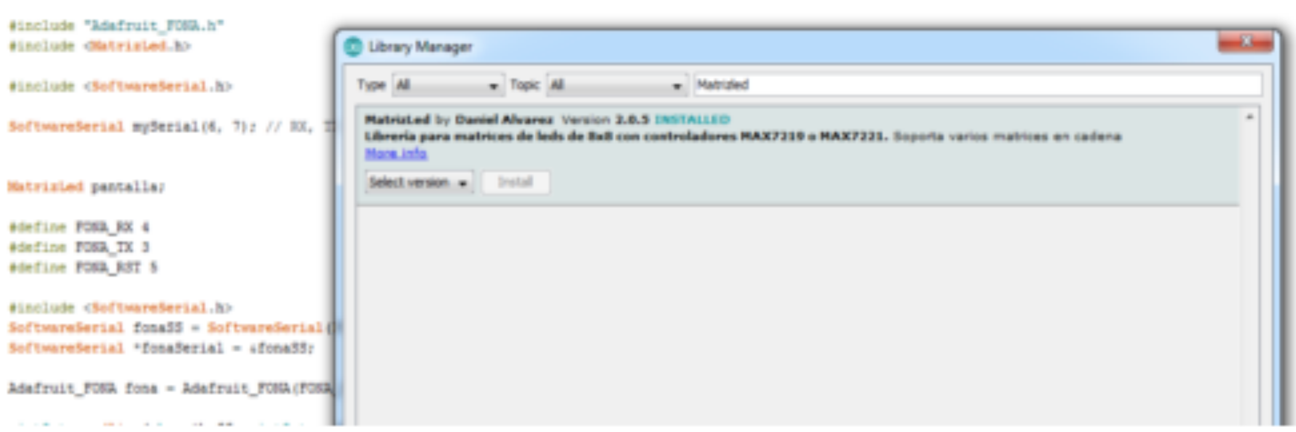
Adafruit FONA Library by **Adafruit** Version **1.3.4** INSTALLED
Arduino library for the Adafruit FONA Arduino library for the Adafruit FONA
More info

Adafruit IO Arduino by Adafruit
Arduino library to access Adafruit IO. Arduino library to access Adafruit IO using the Adafruit AirLift, ESP8266, ESP32, M0 WINC1500, WICED, MKR1000, Ethernet, or FONA hardware.
More info

Adafruit MQTT Library by Adafruit Version 0.20.4 INSTALLED
MQTT library that supports the FONA, ESP8266, Yun, and generic Arduino Client hardware. Simple MQTT library that supports the bare minimum to publish and subscribe to topics.
More info

Ubidots FONA Library by Ubidots Version 1.1.0 INSTALLED
Ubidots library for the Adafruit FONA Ubidots library for the Adafruit FONA
More info

Fig 3.



```
#include "Adafruit_FONA.h"
#include <Matrizled.h>

#include <SoftwareSerial.h>

SoftwareSerial mySerial(6, 7); // RX, T


Matrizled pantalla;

#define FONA_RX 4
#define FONA_TX 3
#define FONA_RST 5

#include <SoftwareSerial.h>
SoftwareSerial fonaSS = SoftwareSerial(
SoftwareSerial *fonaSerial = &fonaSS;

Adafruit_FONA fona = Adafruit_FONA(FONA
```

Library Manager

Type All Topic All Matrizled

MatrizLed by **Daniel Alvarez** Version **2.0.5** INSTALLED
Libreria para matrices de leds de 8x8 con controladores MAX7219 o MAX7221. Soporta varios matrices en cadena
More info

Fig 4.

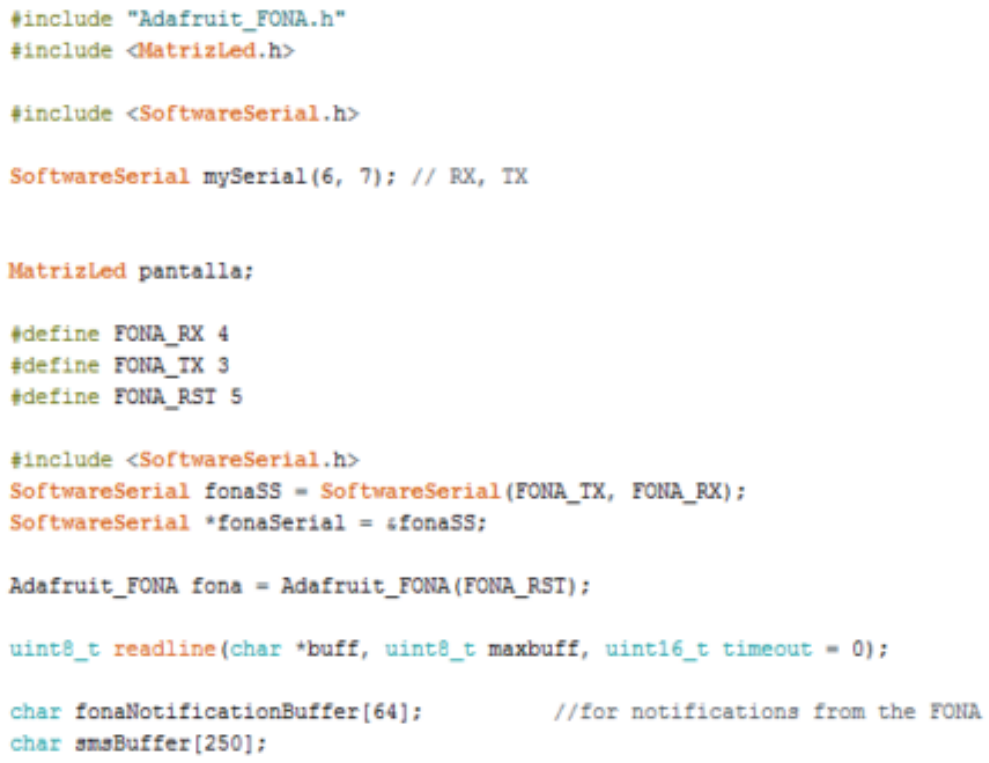
After the successful installation of libraries, we can start coding.

Coding

Firstly, we will have to initialize the required library within the code. Then we will have to create variables to store some values as given in the code snippet (Refer Fig 5.)

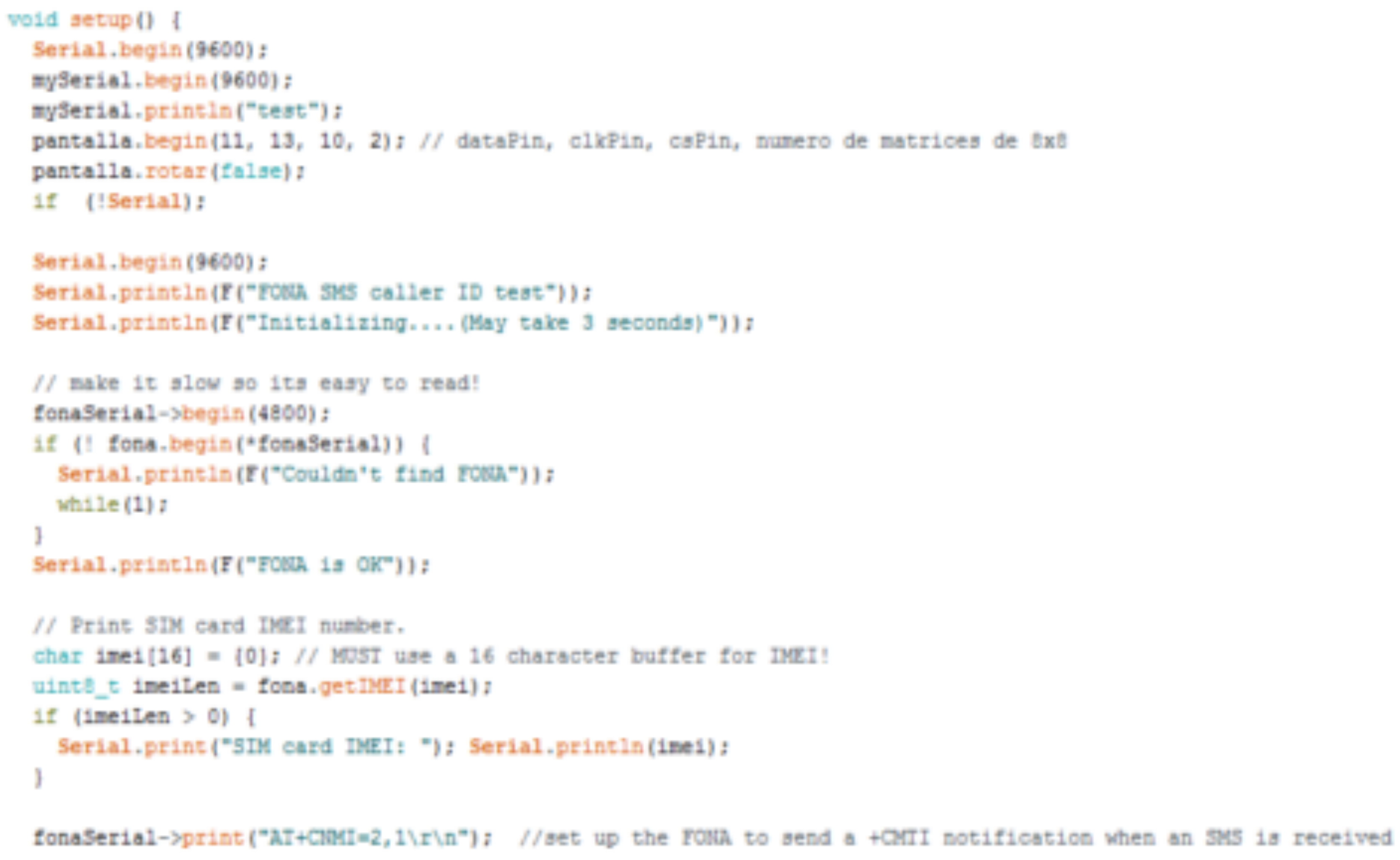
After that, we will create a setup function where we have to set the baud rate = 4800 for serial communication with sim800l module. Then we will create an if condition that will check the connection with sim800l.(Refer Fig 6.)

Now, we have to create a loop function that will continuously check the notification from sim800l module. If a notification from sim800l module is detected about a new SMS, then that will be read and stored in char buffer. After that, we will have to write a line of code that will allow the stored message to be continuously displayed on the LED notice board. (Refer fig 7.)



```
#include "Adafruit_FONA.h"
#include <MatrizLed.h>

#include <SoftwareSerial.h>

SoftwareSerial mySerial(6, 7); // RX, TX


MatrizLed pantalla;

#define FONA_RX 4
#define FONA_TX 3
#define FONA_RST 5

#include <SoftwareSerial.h>
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

Adafruit_FONA fona = Adafruit_FONA(FONA_RST);

uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);

char fonaNotificationBuffer[64];          //for notifications from the FONA
char smsBuffer[250];
```

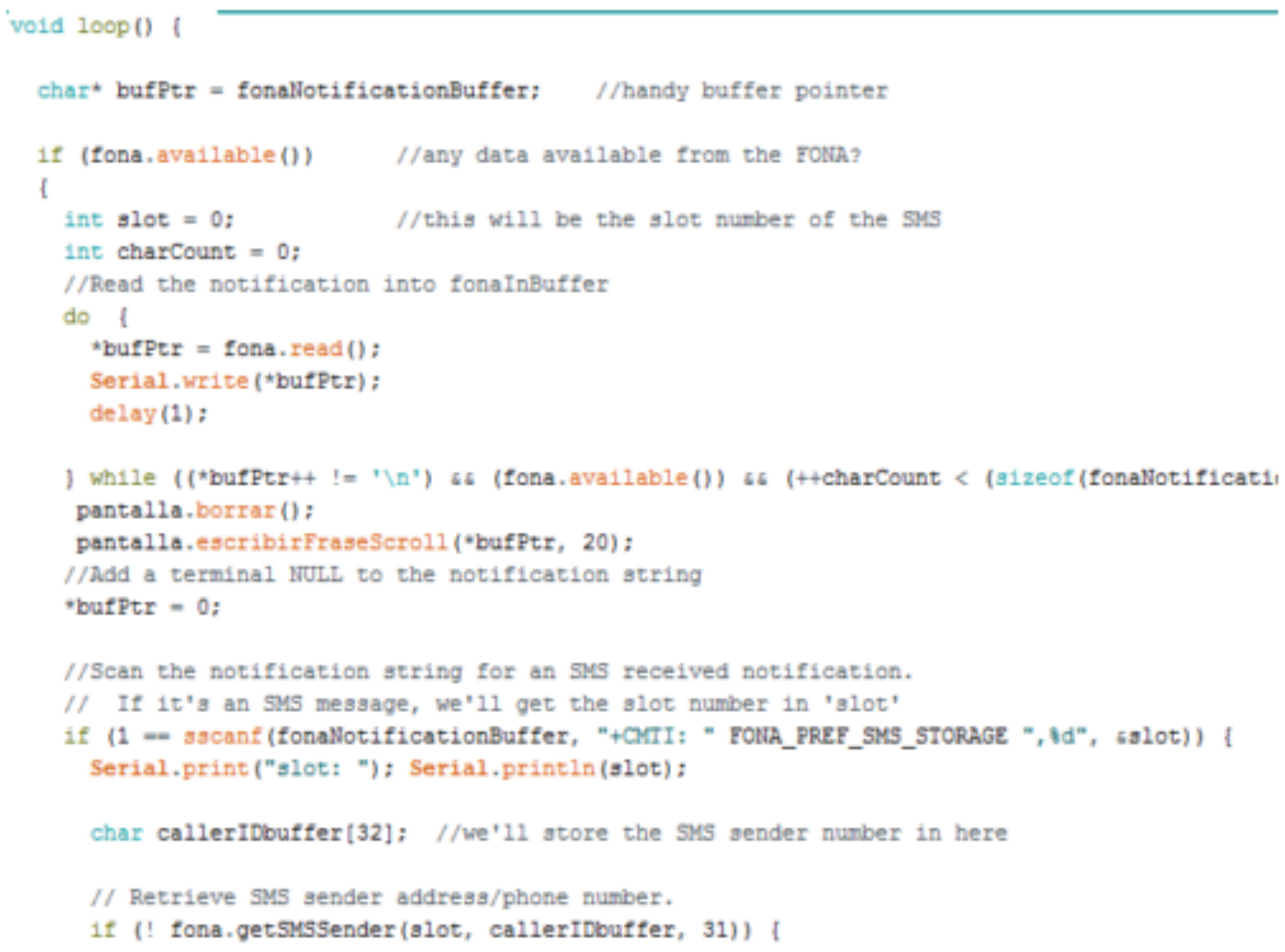
Fig 5



```
void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
  mySerial.println("test");
  pantalla.begin(11, 13, 10, 2); // dataPin, clkPin, csPin, numero de matrices de 8x8
  pantalla.rotar(false);
  if  (!Serial);

  Serial.begin(9600);
  Serial.println(F("FONA SMS caller ID test"));
  Serial.println(F("Initializing....(May take 3 seconds)"));

  // make it slow so its easy to read!
  fonaSerial->begin(4800);
  if (! fona.begin(*fonaSerial)) {
    Serial.println(F("Couldn't find FONA"));
    while(1);
  }
  Serial.println(F("FONA is OK"));

  // Print SIM card IMEI number.
  char imei[16] = {0}; // MUST use a 16 character buffer for IMEI!
  uint8_t imeiLen = fona.getIMEI(imei);
  if (imeiLen > 0) {
    Serial.print("SIM card IMEI: "); Serial.println(imei);
  }

  fonaSerial->print("AT+CNMI=2,1\r\n");  //set up the FONA to send a +CMTI notification when an SMS is received
```

Fig 6

Download the given code, select the right port of the Arduino UNO/Nano/Mini and upload the code to it.



```
void loop() {

  char* bufPtr = fonaNotificationBuffer;    //handy buffer pointer

  if (fona.available())      //any data available from the FONA?
  {
    int slot = 0;            //this will be the slot number of the SMS
    int charCount = 0;
    //Read the notification into fonaInBuffer
    do  {
      *bufPtr = fona.read();
      Serial.write(*bufPtr);
      delay(1);

    } while ((*bufPtr++ != '\n') && (fona.available()) && (++charCount < (sizeof(fonaNotificati
     pantalla.borrar();
     pantalla.escribirFraseScroll(*bufPtr, 20);
    //Add a terminal NULL to the notification string
    *bufPtr = 0;

    //Scan the notification string for an SMS received notification.
    //  If it's an SMS message, we'll get the slot number in 'slot'
    if (1 == sscanf(fonaNotificationBuffer, "+CMTI: " FONA_PREF_SMS_STORAGE ",%d", &slot)) {
      Serial.print("slot: "); Serial.println(slot);

      char callerIDbuffer[32];  //we'll store the SMS sender number in here

      // Retrieve SMS sender address/phone number.
      if (! fona.getSMSSender(slot, callerIDbuffer, 31)) {
```

Fig 7.

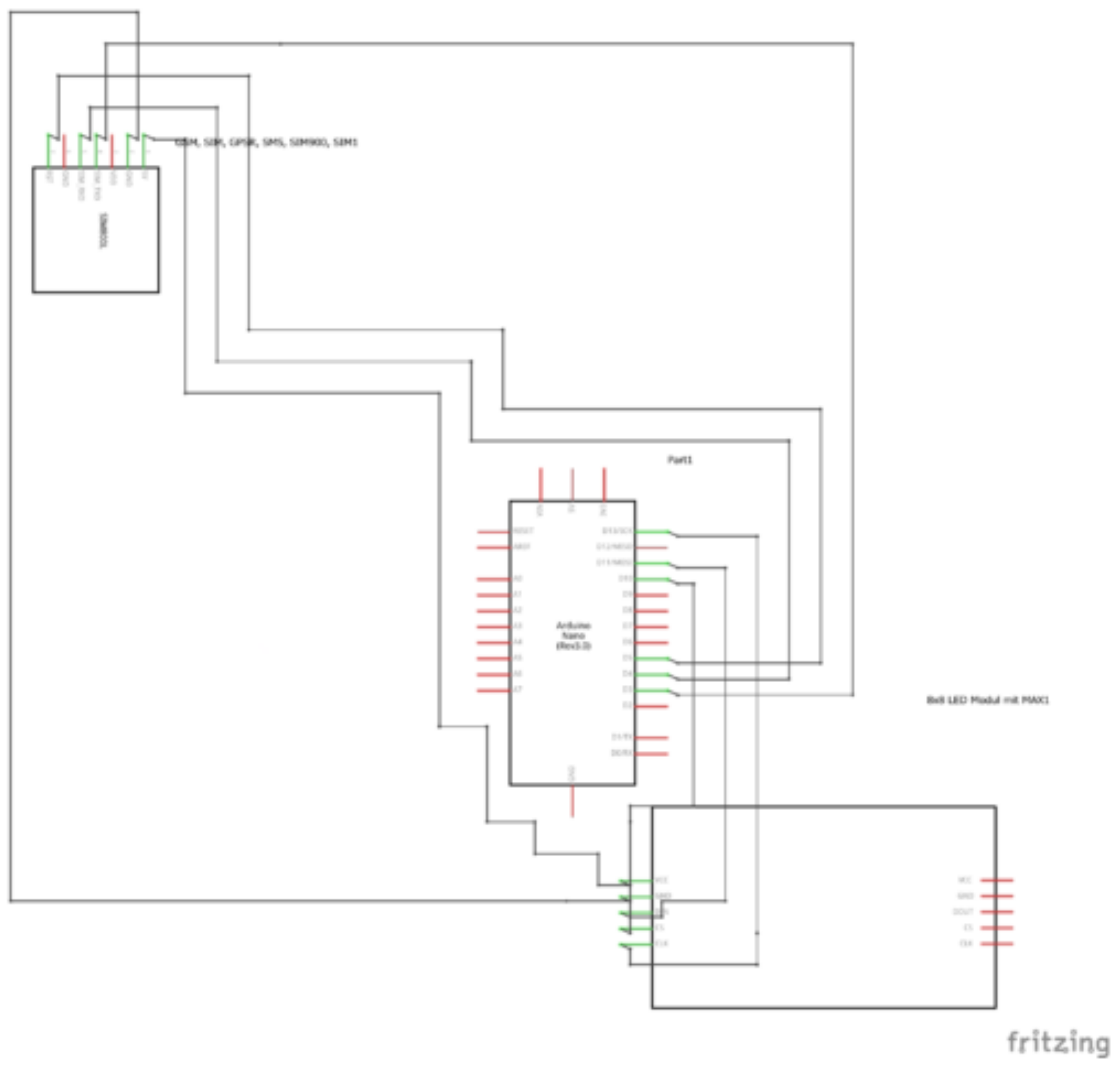


Fig 8 .Circuit diagram

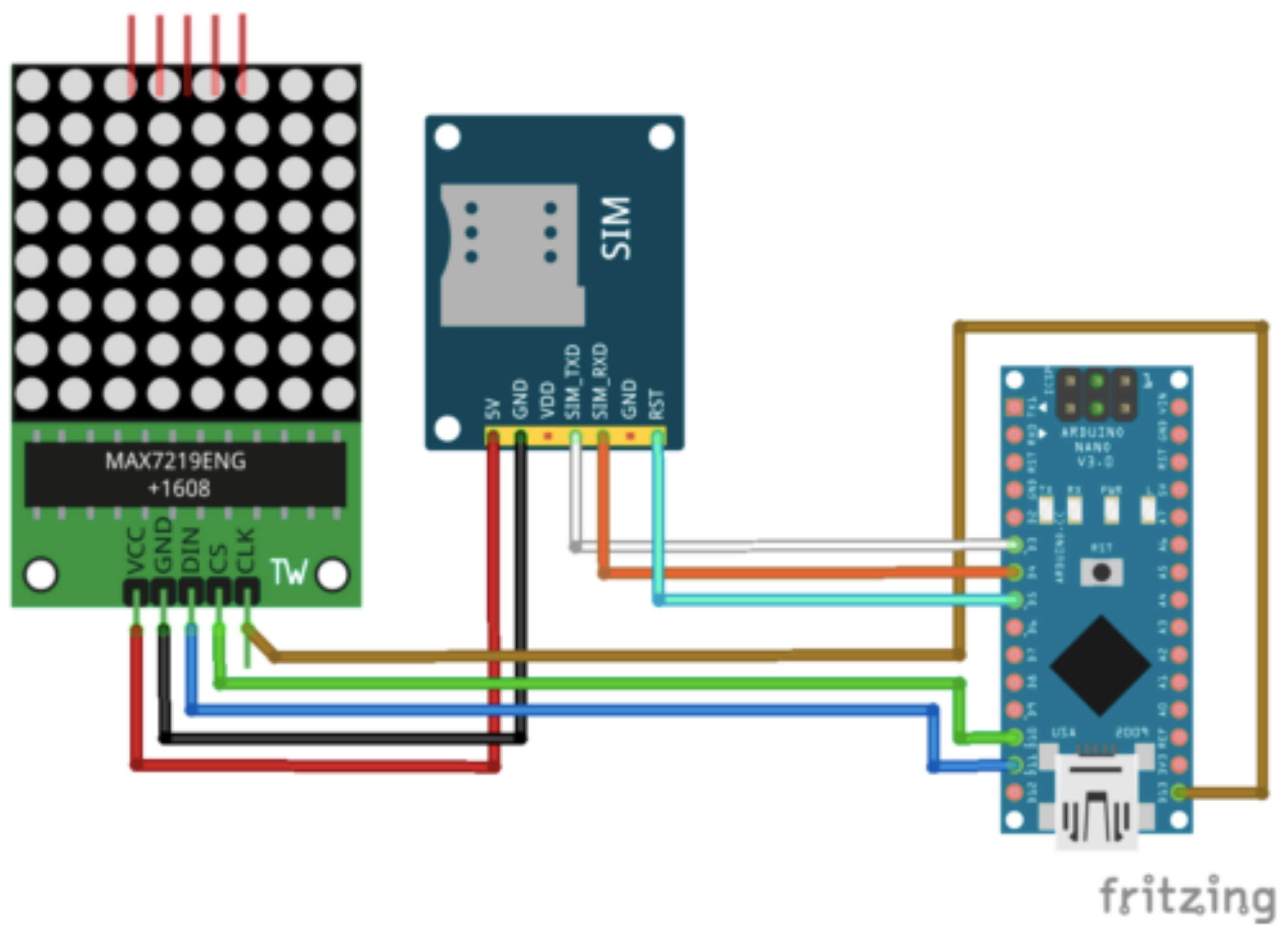


Fig 9.

Now connect the components as described in circuit diagram created in Fritzing(Refer Fig 8,9).

Testing

Now insert the SIM card in Sim800l module (note that the sim operator must have compatibility with 2G Network. I have used an Idea sim. You can use a sim of any other 2G network provider).

Now, power your device using any 5-6V DC battery or adaptor. After powering the module, it will display a message “Send message to number” on the led matrix. Wait a few minutes so that the sim800l can find and connect with the network of the sim. When the Sim800l module connects to the network, the red led on sim 800l module will start blinking with mode delay than previous delay time. Now type a message of your choice and send it to your sim number. The device will immediately display that.

Congrats. Your SMS-based Smart Notice Board is now ready for use.

Download Source Code

You can watch Video for this DIY here: https://youtu.be/R10snQame50

14

PC-BASED OSCILLOSCOPE USING ARDUINO

BY RAMALINGAM BALAJI

Oscilloscopes are an essential tool for electronics hobbyists and professionals to verify that their designs would work as expected. PC based Oscilloscope score over standalone oscilloscopes due to their compact size, low cost and ability to do offline analysis.

Here we describe how you can make your own oscilloscope at a very low cost using your PC and an Arduino board as the hardware for signal acquisition. You can use this oscilloscope to capture frequency signals up to 5kHz. The Arduino board, the heart of the oscilloscope, reads the values from its inbuilt analogue-to-digital converter (ADC) and pushes these to the PC via USB port. We have provided here an Arduino sketch, which you can compile and load directly to the Arduino. You also need to install an executable file or application in your Windows PC. This application works as the front-end to plot input signals as waveforms on your computer screen.

The Arduino board consists of Atmel's AVR microcontroller, which can be 8-, 16- or 32-bit based on the type of the board. For this project, you can use any variant of the Arduino as hardware. The AVR microcontroller has an inbuilt ADC. In the project, we use pin A0 to capture the input signal. The captured input signal is fed to UART via UART-USB converter in the Arduino to the PC. A virtual COM port is created by Windows whenever the Arduino connects to the PC. A Windows-based application developed using NI LabWindows opens up the virtual COM port and starts plotting signals visually using Graph libraries.

The sampling speed of the oscilloscope is limited by the baud rate of the UART. The Arduino sketch is coded to read the ADC using ISR, and the UART baud rate is configured at 115200, which sends data at 85µs intervals. This gives an effective sampling rate of 12kSa/s.

Construction

The PC scope set-up is quite simple and straightforward as shown in Fig. 1. The Arduino board connects to your laptop or PC via the USB cable. Any external power supply for the board is not required as the board is powered by the USB only. Connect switching diodes (D1 and D2) as input protection circuit to pin A0 of the Arduino's ADC. You need Arduino sketch (pcscope.ino) and PC software or executable file (PCScope.exe) in order to use this circuit. Install PCScope.exe program (developed by author) in your Windows PC and open the application. Next, open the Arduino sketch from Arduino IDE and compile the sketch. Connect the Arduino board to the PC and flash the sketch into the microcontroller on the Arduino board.

The ADC of Arduino can measure voltages up to 5V. So it is advisable to add a small protection circuit to limit the input voltage to 5V and clamp the negative voltage.

A low-power, fast-switching diode like 1N4148 can be used to protect the input pin. Connect a 10-kilo-ohm resistor in series with the input. It will work as a current limiter in case the input goes beyond 5V. Additional voltage dividers can be used in case you need to measure voltages higher than 5V.

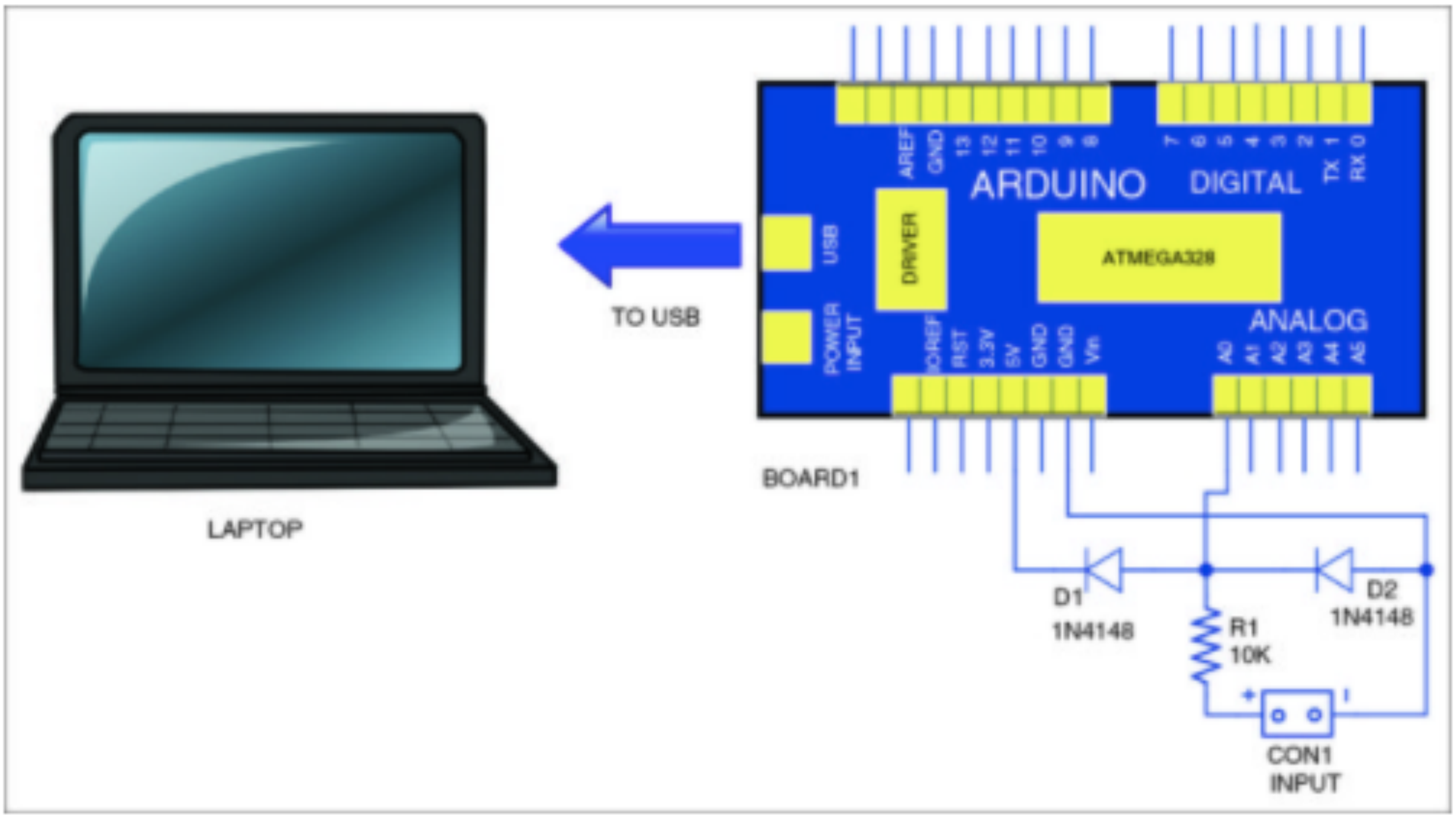


Fig. 1: Circuit of the PC-based oscilloscope using Arduino

‘

Software

Arduino sketch. The sampling rate of this PC scope application is limited by the rate at which the data is sent to the PC. Baud rate of 115000 gives time interval of around 85 µs. It is important to get the ADC signals much before this time to get reliable data plotting. The sketch reads pin A0 of Board1 and sends to UART at 115200 baud rate. At this speed, bytes of the input are pushed at time intervals of around 85µs.

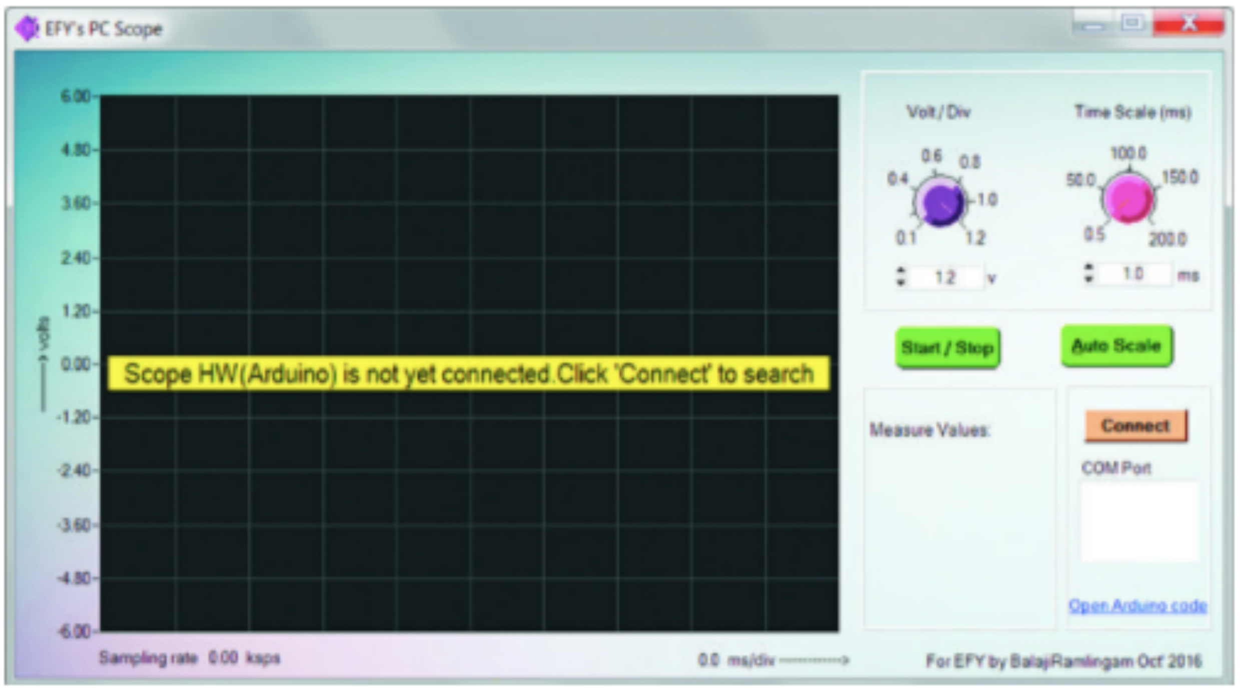
By default, the ADC configuration of the Arduino gives samples every 116µs. So here the ADC is configured with additional lines of code to get samples faster than 85µs by setting the prescaler to 16. With this, you get ADC conversion every 20µs, which is much faster than the UART data transfer rate.

Download source code

PC software. As stated earlier, the front-end PC software for signal acquisition and processing is developed using NI LabWindows. The serial port data is captured through Arduino at regular time intervals and plotted as a graph on the screen using the Plot function library. The display points along X-axis are calculated based on the user-defined time scale. The Y-axis range is set using the voltage selection control.

Testing

After installing the PC scope application, click 'Connect' button on your PC screen to connect to the Arduino board (Fig. 2). When the board gets connected to your PC, you will get a confirmation message for three seconds as shown in Fig. 3.



Scope HW(Arduino) is not yet connected.Click 'Connect' to search

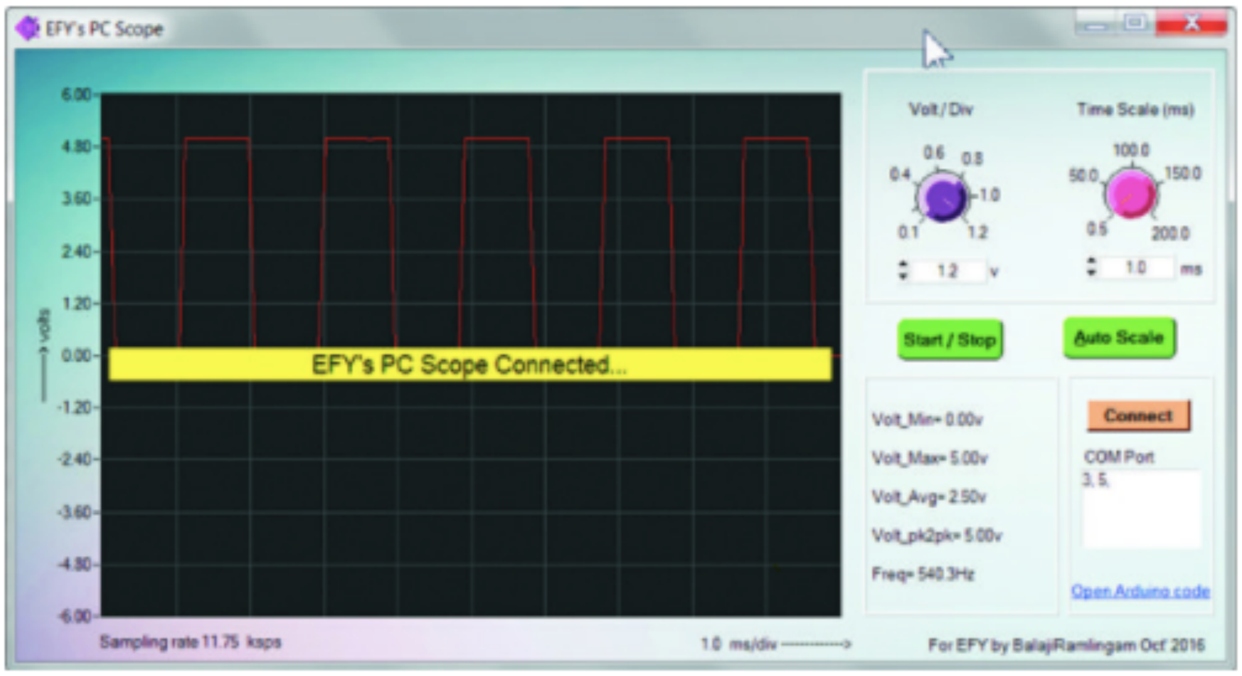
Measure Values:

Sampling rate 0.00 ksps

0.0 ms/div

For EFY by BalajiRamlingam Oct 2016

Fig. 2: Message on the screen when the PC-based scope is run for the first time



EFY's PC Scope Connected...

Volt_Min= 0.00v
Volt_Max= 5.00v
Volt_Avg= 2.50v
Volt_pk2pk= 5.00v
Freq= 540.3Hz

COM Port 3 5

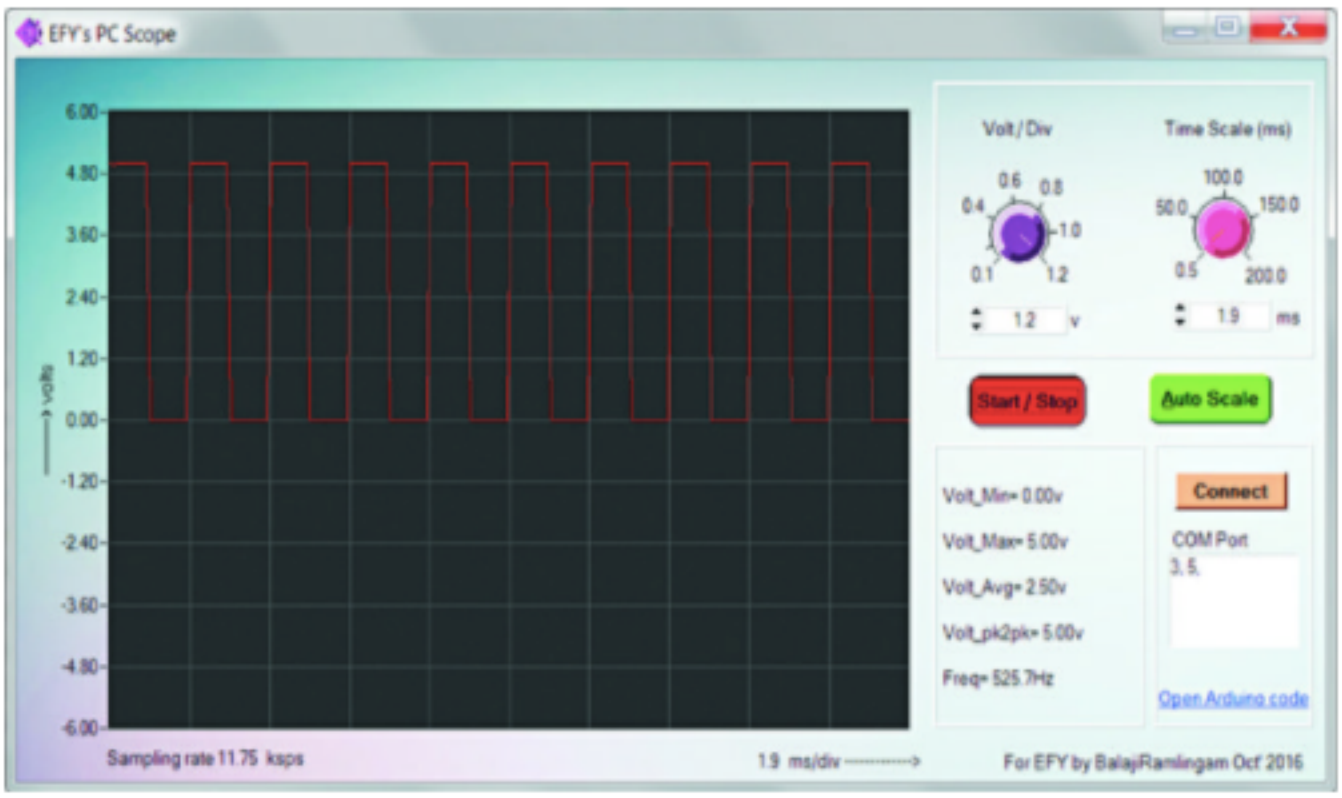
Sampling rate 11.75 ksps

1.0 ms/div

For EFY by BalajiRamlingam Oct 2016

Fig. 3: Message after the hardware successfully connects to the PC

Feed any squarewave input of up to 5kHz at CON1. The software must plot its output waveform on your PC. Square and triangular output waveforms of 525Hz and 530Hz captured on the screen during testing are shown in Figs 4 and 5, respectively. Similarly, you can feed rectangular or pulse inputs (but not sine waves) to get output waveforms.



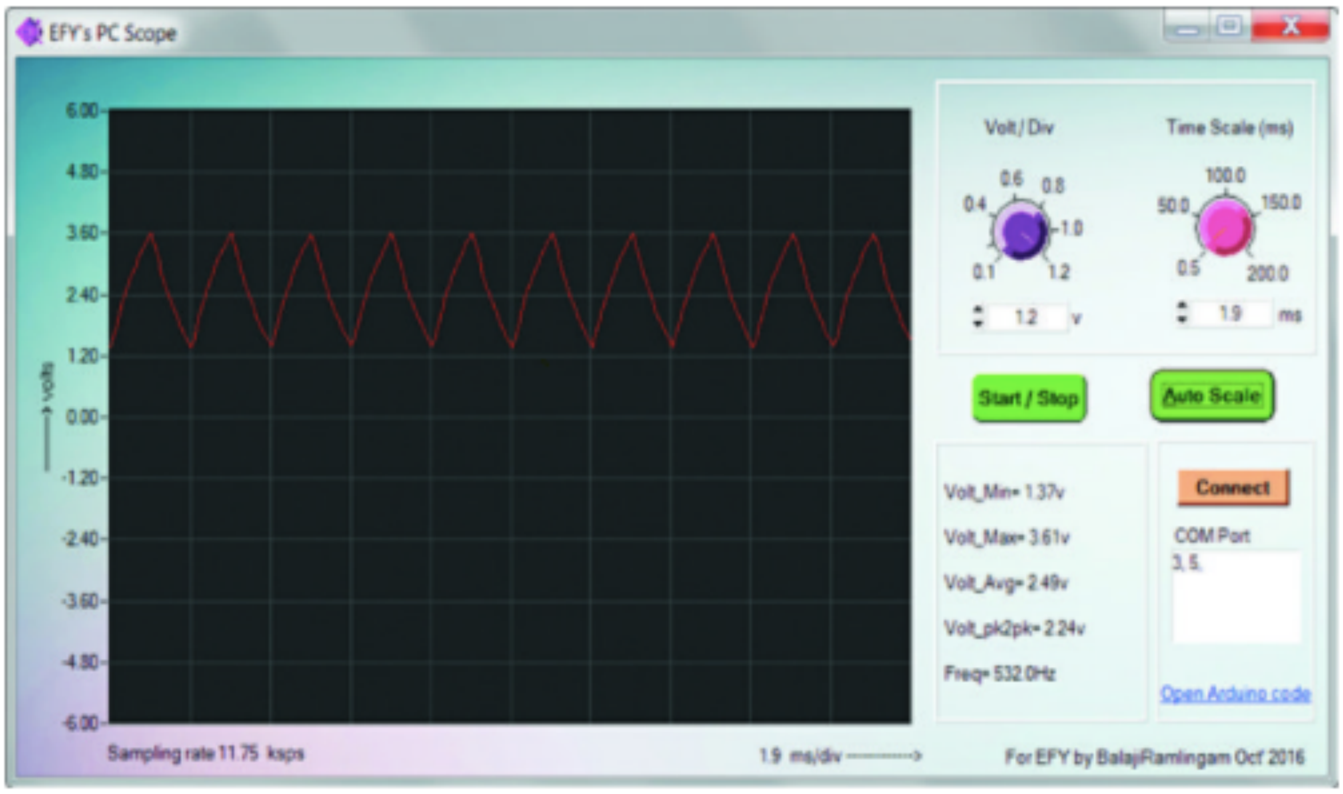
Volt/Div: 1.2 v
Time Scale: 1.9 ms

Volt_Min= 0.00v
Volt_Max= 5.00v
Volt_Avg= 2.50v
Volt_pk2pk= 5.00v
Freq= 525.7Hz

COM Port 3, 5,

Sampling rate 11.75 ksps   1.9 ms/div ------>   For EFY by BalajiRamlingam Oct 2016

Fig. 4: Test signal of 525Hz square waveform captured on the screen



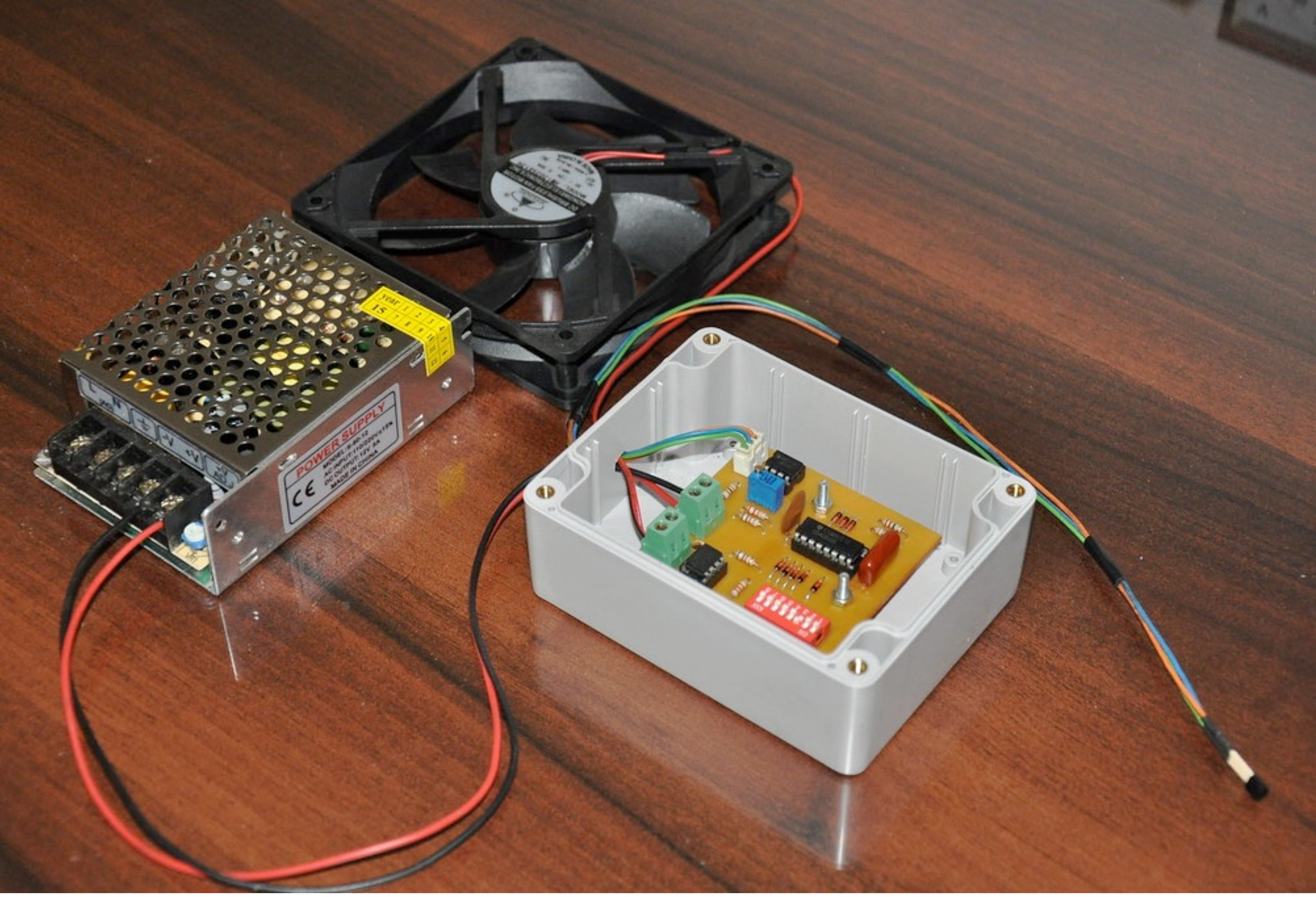
Volt/Div: 1.2 v
Time Scale: 1.9 ms

Volt_Min= 1.37v
Volt_Max= 3.61v
Volt_Avg= 2.49v
Volt_pk2pk= 2.24v
Freq= 532.0Hz

COM Port 3, 5,

Sampling rate 11.75 ksps   1.9 ms/div ------>   For EFY by BalajiRamlingam Oct 2016

Fig. 5: Test signal of 530Hz triangular waveform captured on the screen



15

TEMPERATURE BASED FAN SPEED CONTROL AND MONITORING USING ARDUINO

BY BISWAJIT DAS

This project is a standalone automatic fan speed controller that controls the speed of an electric fan according to the requirement. Use of embedded technology makes this closed-loop feedback-control system efficient and reliable. The microcontroller (MCU) ATMega8/168/328 allows dynamic and faster control and the LCD makes the system user-friendly. Sensed temperature and fan speed levels are simultaneously displayed on the LCD panel.

The project is very compact and uses a few components only. It can be implemented for several applications including air-conditioners, water-heaters, snow-melters, ovens, heat-exchangers, mixers, furnaces, incubators, thermal baths and veterinary operating tables. The project will help save energy/electricity.

Circuit and working

Circuit diagram of the temperature fan speed control and monitoring is shown in Fig. 1. It is built around Arduino Uno board (Board1), 16x2 LCD (LCD1), temperature sensor LM35 (IC1) and a few other components.

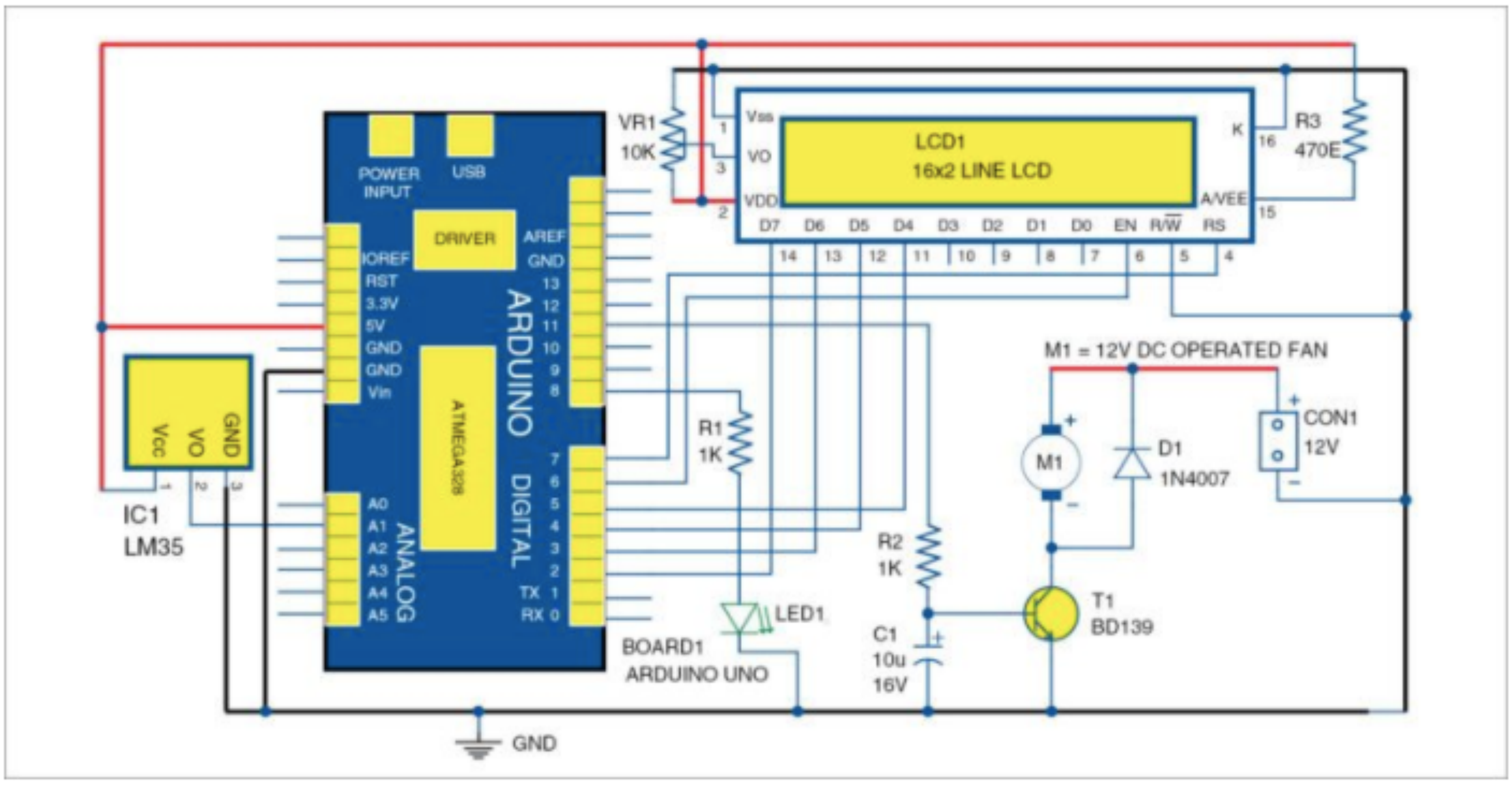
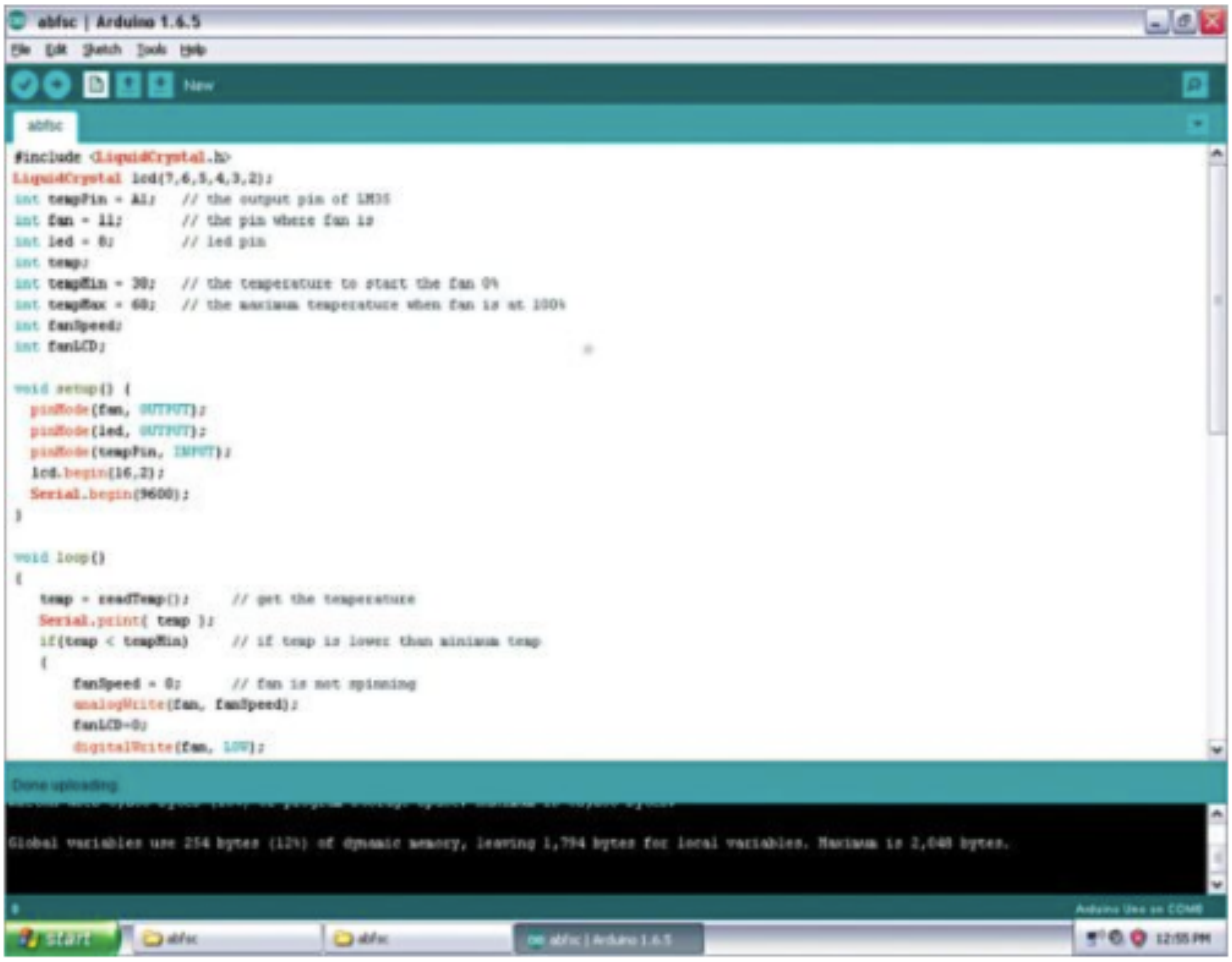


Fig. 1: Circuit diagram of the temperature-based fan speed control and monitoring using Arduino

Arduino is at the heart of the circuit as it controls all functions. LM35 is a precision integrated circuit whose output voltage is linearly proportional to Celsius (Centigrade) temperature. It is rated to operate over a -55°C to 150°C temperature range. It has +10.0mV/Celsius linear-scale factor.

Temperature sensor LM35 senses the temperature and converts it into an electrical (analogue) signal, which is applied to the MCU through an analogue-to-digital converter (ADC). The analogue signal is converted into digital format by the ADC. Sensed values of the temperature and speed of the fan are



```
#include <LiquidCrystal.h>
LiquidCrystal lcd(7,6,5,4,3,2);
int tempPin = A1;   // the output pin of LM35
int fan = 11;       // the pin where fan is
int led = 8;        // led pin
int temp;
int tempMin = 30;   // the temperature to start the fan 0%
int tempMax = 60;   // the maximum temperature when fan is at 100%
int fanSpeed;
int fanLCD;

void setup() {
  pinMode(fan, OUTPUT);
  pinMode(led, OUTPUT);
  pinMode(tempPin, INPUT);
  lcd.begin(16,2);
  Serial.begin(9600);
}

void loop()
{
   temp = readTemp();     // get the temperature
   Serial.print( temp );
   if(temp < tempMin)     // if temp is lower than minimum temp
   {
       fanSpeed = 0;      // fan is not spinning
       analogWrite(fan, fanSpeed);
       fanLCD=0;
       digitalWrite(fan, LOW);
```

```
Global variables use 254 bytes (12%) of dynamic memory, leaving 1,794 bytes for local variables. Maximum is 2,048 bytes.
```

Fig. 2: Screenshot of the source code on Arduino IDE

displayed on the LCD. Temperature and monitoring using Arduino The MCU on Arduino drive the motor driver to control fan speed.

PARTS LIST

Semiconductors:	
Board1	- Arduino Uno
LCD1	- 16x2 LCD
IC1	- LM35 temperature sensor
T1	- BD139 npn transistor
D1	- 1N4007 rectifier diode
LED1	- 5mm LED
Resistors (all 1/4-watt, ±5% carbon):	
R1, R2	- 1-kilo-ohm
R3	- 470-ohm
VR1	- 10-kilo-ohm preset
Capacitor:	
C1	- 10µF, 16V electrolytic
Miscellaneous:	
CON1	- 2-pin terminal connector
CON2	- 3-pin connector
CON3	- 8-pin connector
M1	- 12V DC operated fan
	- 12V battery for fan

Fan speed control technique

A low-frequency pulse-width modulation (PWM) signal, usually in the range of about 30Hz, whose duty cycle is varied to adjust the fan's speed is used. An inexpensive, single, small pass transistor can be used here. It is efficient because the pass transistor is used as a switch.

One disadvantage of this approach,

however, is that it can make the fan noisy because of the pulsed nature of the signal. The PWM waveform's sharp edges cause the fan's mechanical structure to move (like a badly-designed loudspeaker), which can easily be audible.

Construction and testing

A single-side PCB for the temperature-based fan speed control and monitoring circuit is shown in Fig. 3 and its component layout in Fig. 4. Assemble the circuit on the PCB.

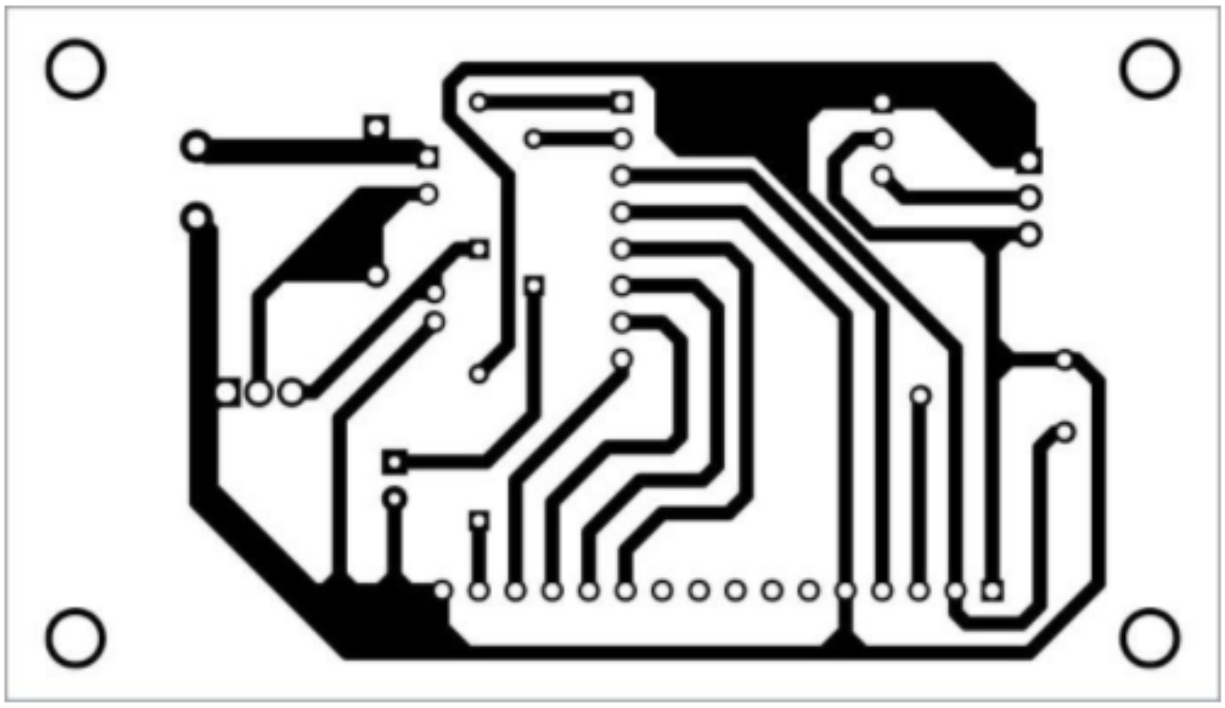


Fig. 3: Actual-size PCB pattern of the temperature-based fan speed control and monitoring circuit using Arduino

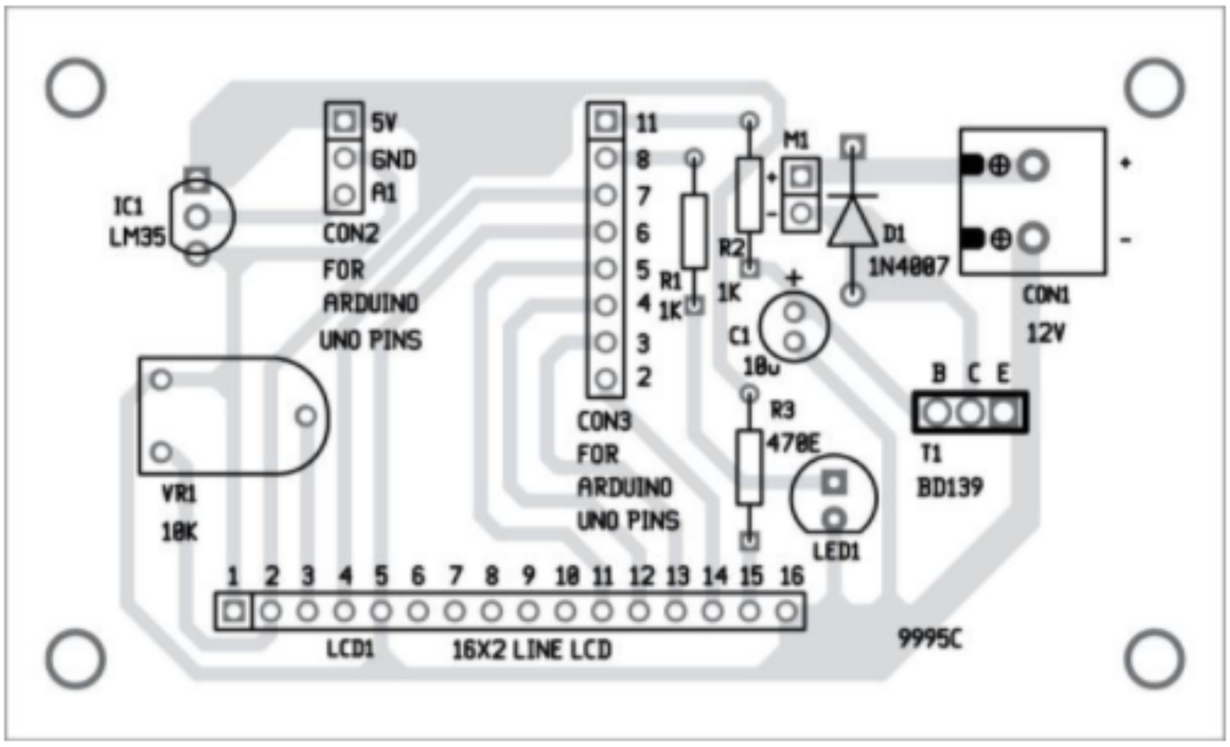


Fig. 4: Component layout of the PCB

CON2 and CON3 are used to connect Board1 (Arduino UNO board) through external connectors. A 12V battery is used to drive the 12V DC-operated fan.

Software

Software for the automatic temperature controller and monitor circuit is written in Arduino programming language. Arduino Uno is programmed using Arduino IDE software.

ATmega328P on Arduino Uno comes with a pre-programmed bootloader that allows users to upload a new code to it without using an external hardware programmer.

Connect Arduino board to the PC and select the correct COM port in Arduino IDE. Compile the program (sketch). Then select the correct board from Tools Board menu in Arduino IDE and upload the sketch (abfc.ino) to Arduino through standard USB port.

Download PCB and Component Layout PDFs: click here

Download the source code: click here

16

SMART WIRELESS WATER METER WITH WEB DB IOT PROJECTS

BY ASHWINI KUMAR SINHA

As water consumption and wastage increases day by day, water scarcity has become a growing concern around the globe. While some irresponsible people are overusing water and wasting it daily, a large population remain deprived of safe drinking water. This problem is caused by the irregular distribution of water.

To deal with this, we need a solution that gives data about daily and monthly water consumption of each area and home to the government to enable proper water distribution. Good news is we have the solution now.

In this project, we are going to make a smart water meter that measures our daily water usage and provides live data on our phone that is connected to a database, which can be accessed by the government.

To start the project, we would require some components.

Components Required:

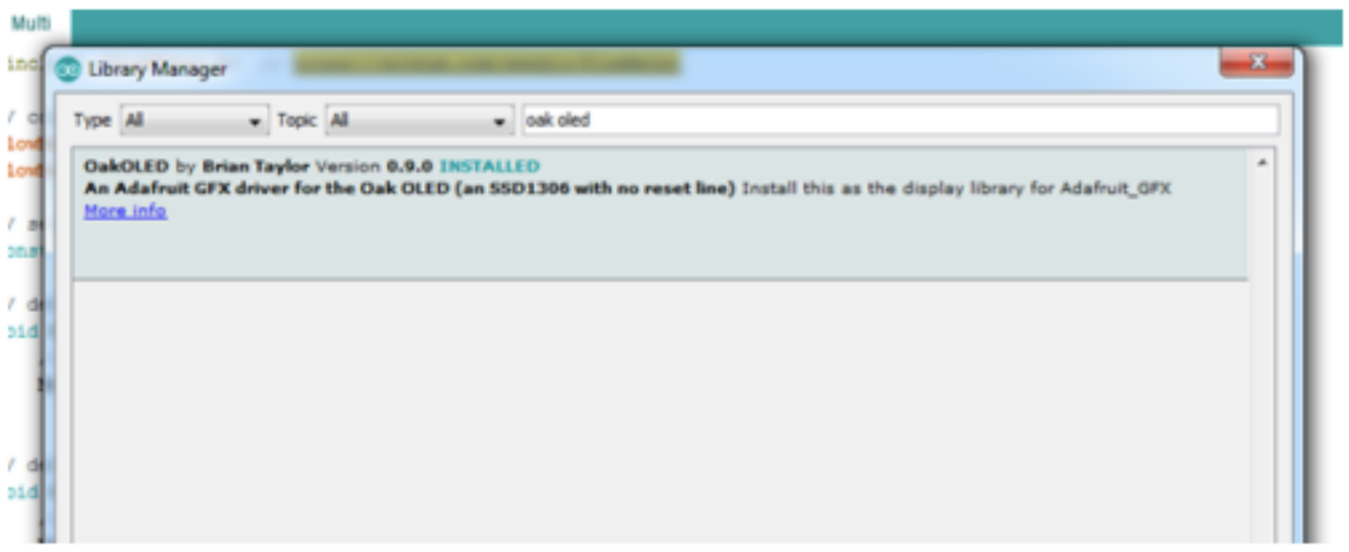
- Water flow meter(Recommended FS400).
- Arduino pro mini(5V and 16 MHz)
- Bluetooth HC 05
- Programmer (Arduino UNO/FTDI programmer)
- OLED SSD 1306
- Water Pipe(according to your sensor inlet diameter)
- Water to tap connector

Once we have collected these components, we can start our project.

Setting Arduino IDE

Download the flow meter library and paste it in libraries folder of Arduino

After that open Arduino and click on manage library and search Oak SSD 1306 OLED library as in the pic below (Fig.1).



OakOLED by Brian Taylor Version 0.9.0 INSTALLED
An Adafruit GFX driver for the Oak OLED (an SSD1306 with no reset line) Install this as the display library for Adafruit_GFX

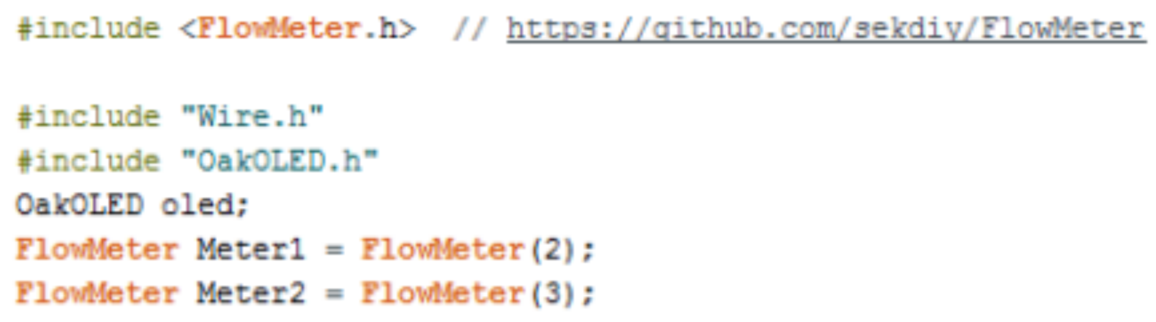
Fig. 1 Installing library

Now our Arduino IDE is ready. Let us start coding part.

Coding:

Open flow meter multi example code and add some of these parts.

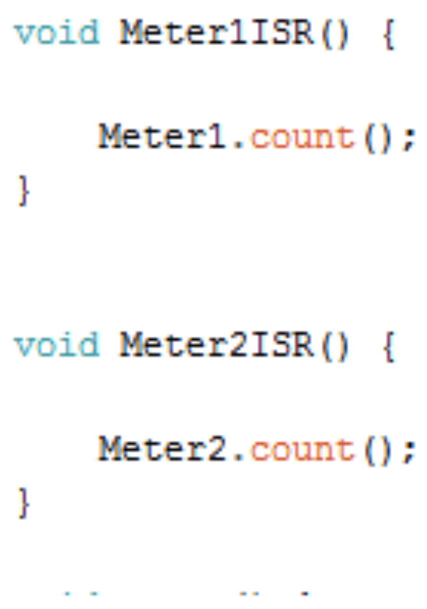
First, we include libraries of flow meter and OLED display as in (Fig. 2).



```
#include <FlowMeter.h>  // https://github.com/sekdiy/FlowMeter

#include "Wire.h"
#include "OakOLED.h"
OakOLED oled;
FlowMeter Meter1 = FlowMeter(2);
FlowMeter Meter2 = FlowMeter(3);
```

Fig 2. Arduino code Including library

After this, we create the interrupt handler for the sensor as in (Fig. 3).

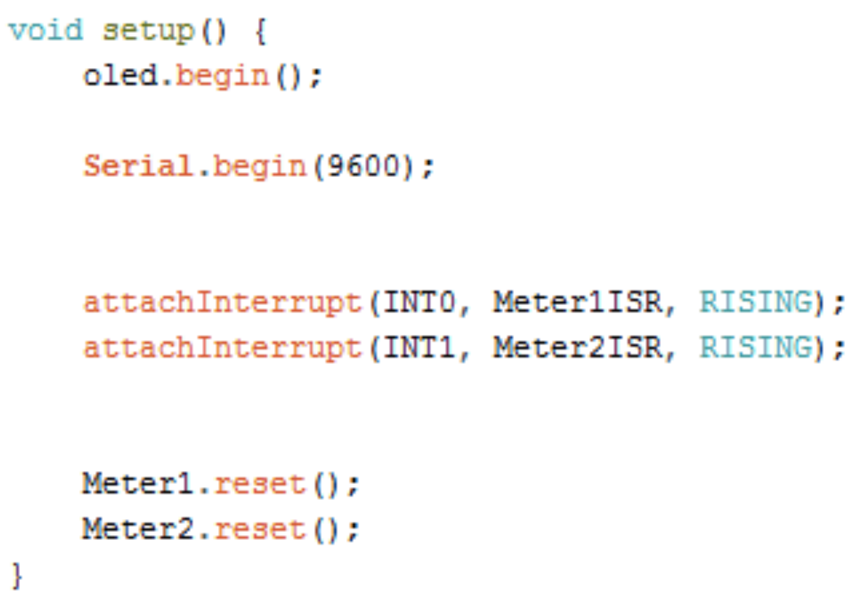


```
void Meter1ISR() {

    Meter1.count();
}


void Meter2ISR() {

    Meter2.count();
}
```

Fig 3. Setting Interrupt

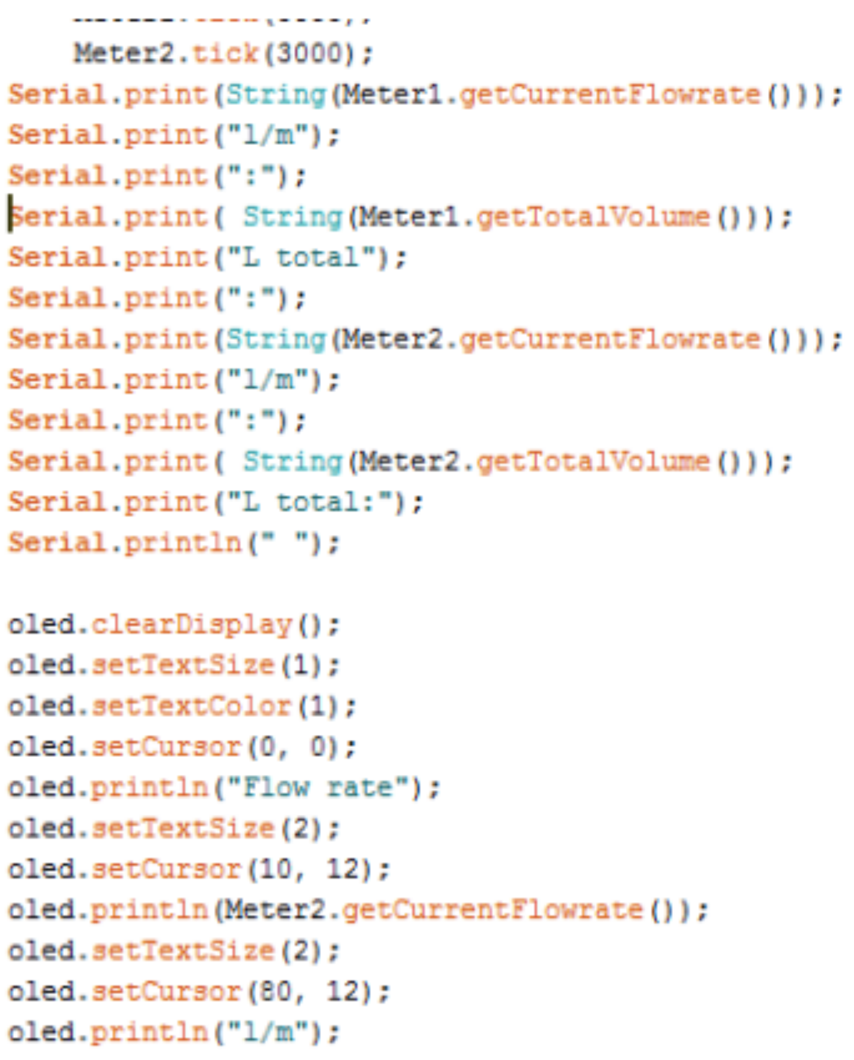
Here, we set up the Bluetooth with Baudrate 9600. If it doesn't work, try default Baudrate 34800.

Finally, we set the data output that will send to Bluetooth with Serial.print(data to send).



```
void setup() {
    oled.begin();

    Serial.begin(9600);


    attachInterrupt(INT0, Meter1ISR, RISING);
    attachInterrupt(INT1, Meter2ISR, RISING);


    Meter1.reset();
    Meter2.reset();
}
```

Fig 4. Setting Bluetooth



```
    Meter2.tick(3000);
Serial.print(String(Meter1.getCurrentFlowrate()));
Serial.print("l/m");
Serial.print(":");
Serial.print( String(Meter1.getTotalVolume()));
Serial.print("L total");
Serial.print(":");
Serial.print(String(Meter2.getCurrentFlowrate()));
Serial.print("l/m");
Serial.print(":");
Serial.print( String(Meter2.getTotalVolume()));
Serial.print("L total:");
Serial.println(" ");

oled.clearDisplay();
oled.setTextSize(1);
oled.setTextColor(1);
oled.setCursor(0, 0);
oled.println("Flow rate");
oled.setTextSize(2);
oled.setCursor(10, 12);
oled.println(Meter2.getCurrentFlowrate());
oled.setTextSize(2);
oled.setCursor(80, 12);
oled.println("l/m");
```

Fig 5. Display data to OLED

Then set OLED display function to display data on OLED screen with oled.println(data to send) refer (Fig 5)

As we are now done with the coding part, next is app making:

App Making

Open MIT App Inventor, make a layout and add the components as in Fig 6.

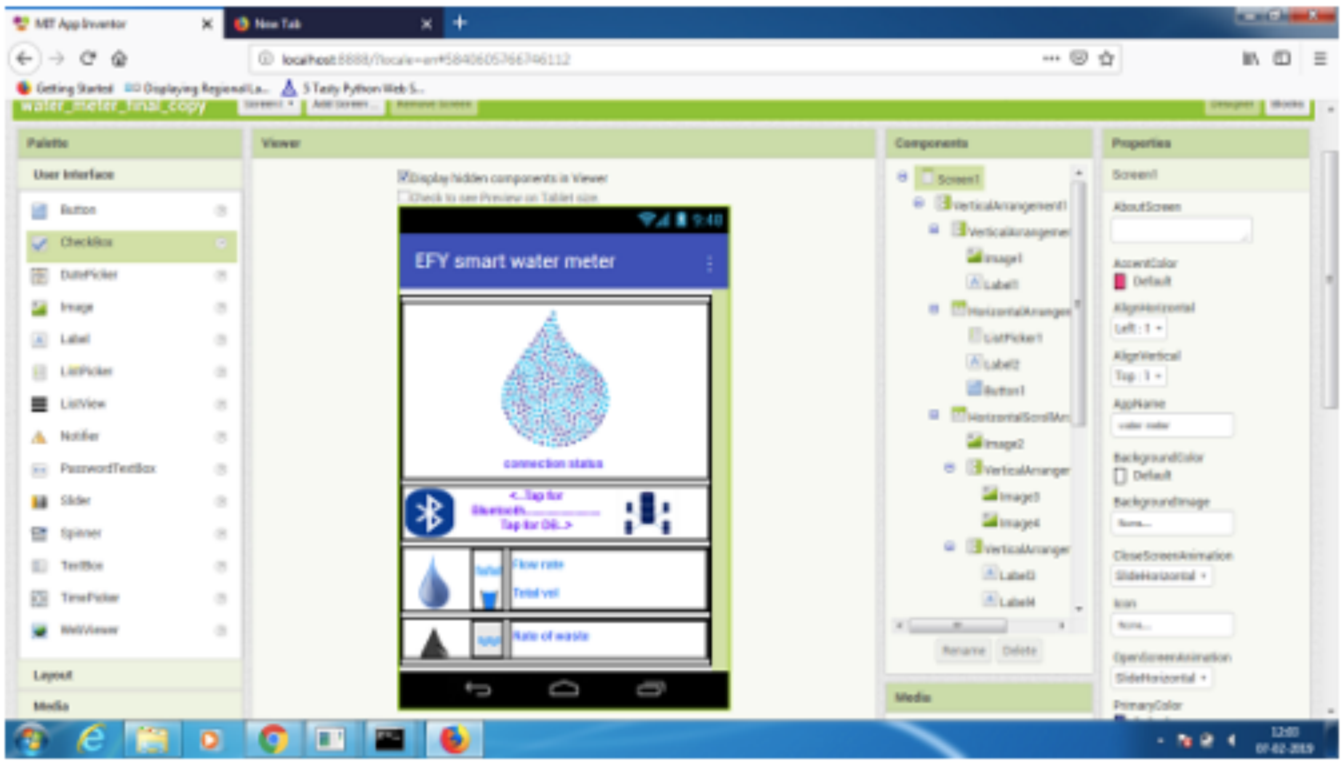


Fig 6. App layout

Main Components we added in the layout are:

- 1 Buttons one for DB.
- 5 Text view for Bluetooth connection status, Water flow rate of inlet, Total volume of water consumed, the flow rate of wastewater and total volume of water waste.
- Tiny web DB
- List picker for Bluetooth
- Bluetooth client
- Timer clock

Now go to MIT Code block and set the code as in Fig 7.

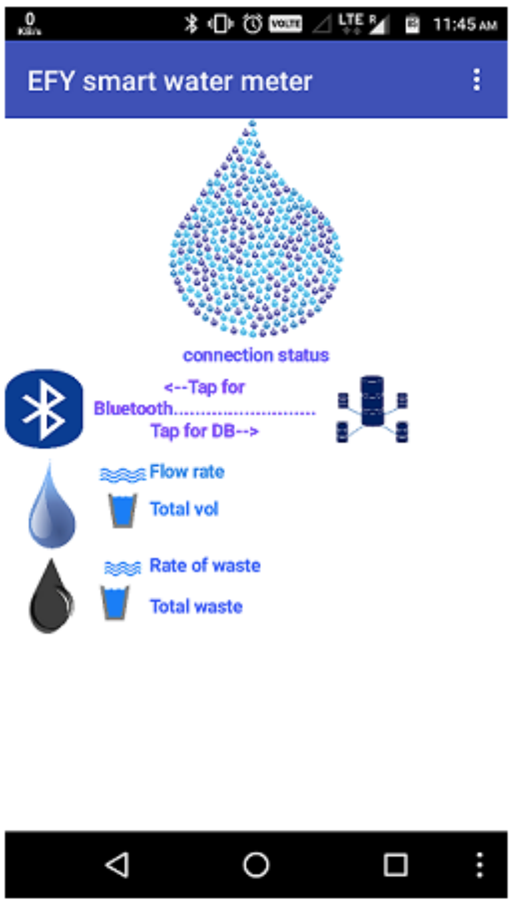
You can download the whole code and app from link below with .aia file.

Code Water Meter

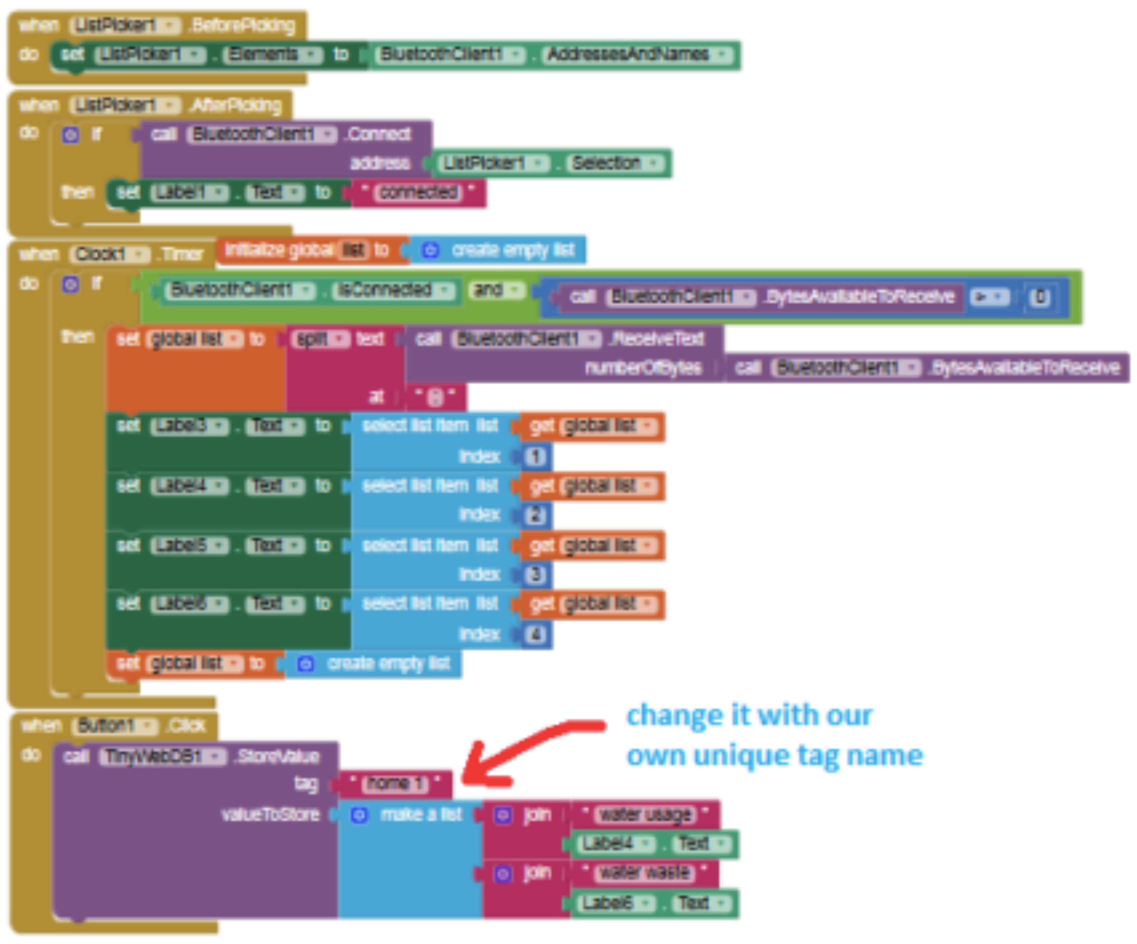
[caption id="attachment_52085" align="aligncenter" width="500"] FIG 7. MIT APP Inventor Coding[/caption]

We have made APP and we have finished the code. What we need now is to upload the code to Arduino.

NOTE: I have used Arduino pro mini to make the project small and easy, you can also use Arduino UNO for this project. Code and connection remain same for both, it's all about your choice.



App Layout



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  if   call BluetoothClient1 .Connect
                               address  ListPicker1 . Selection
    then set Label1 . Text to " connected "

initialize global list to create empty list

when Clock1 .Timer
do  if   BluetoothClient1 . IsConnected and call BluetoothClient1 .BytesAvailableToReceive > 0
    then set global list to split text call BluetoothClient1 .ReceiveText
                                                   numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
                                   at " @ "
         set Label3 . Text to select list item list get global list
                                               index 1
         set Label4 . Text to select list item list get global list
                                               index 2
         set Label5 . Text to select list item list get global list
                                               index 3
         set Label6 . Text to select list item list get global list
                                               index 4
         set global list to create empty list

when Button1 .Click
do  call TinyWebDB1 .StoreValue
                   tag " home 1 "
          valueToStore make a list  join " water usage "
                                         Label4 . Text
                                    join " water waste "
                                         Label6 . Text
```

FIG 7. MIT APP Inventor Coding

Uploading Code To Arduino Pro Mini:

First remove Arduino UNO IC as in (Fig. 8).

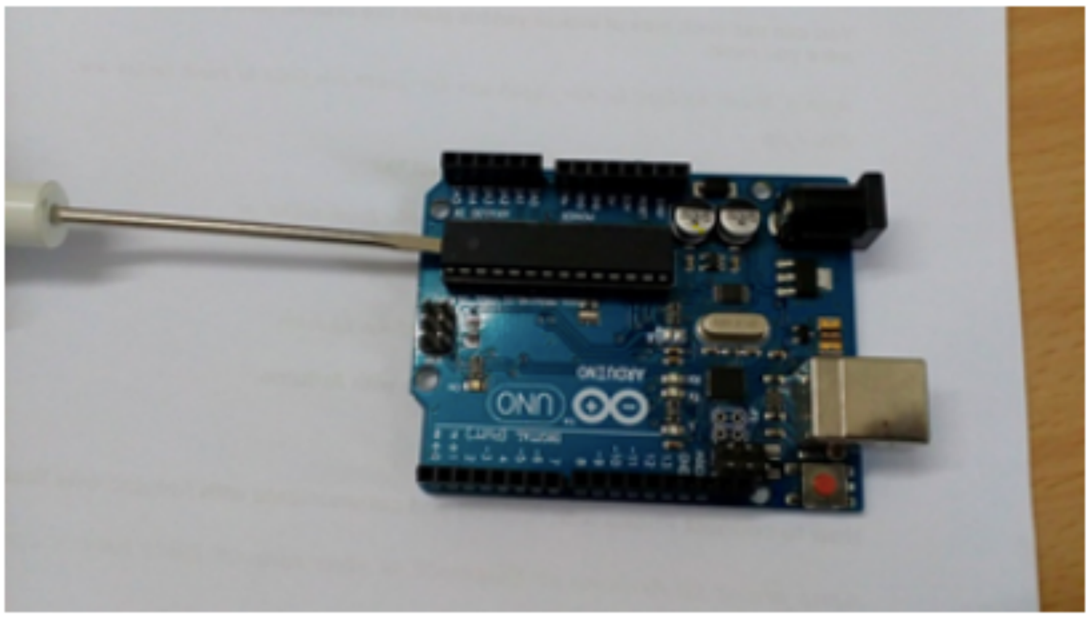
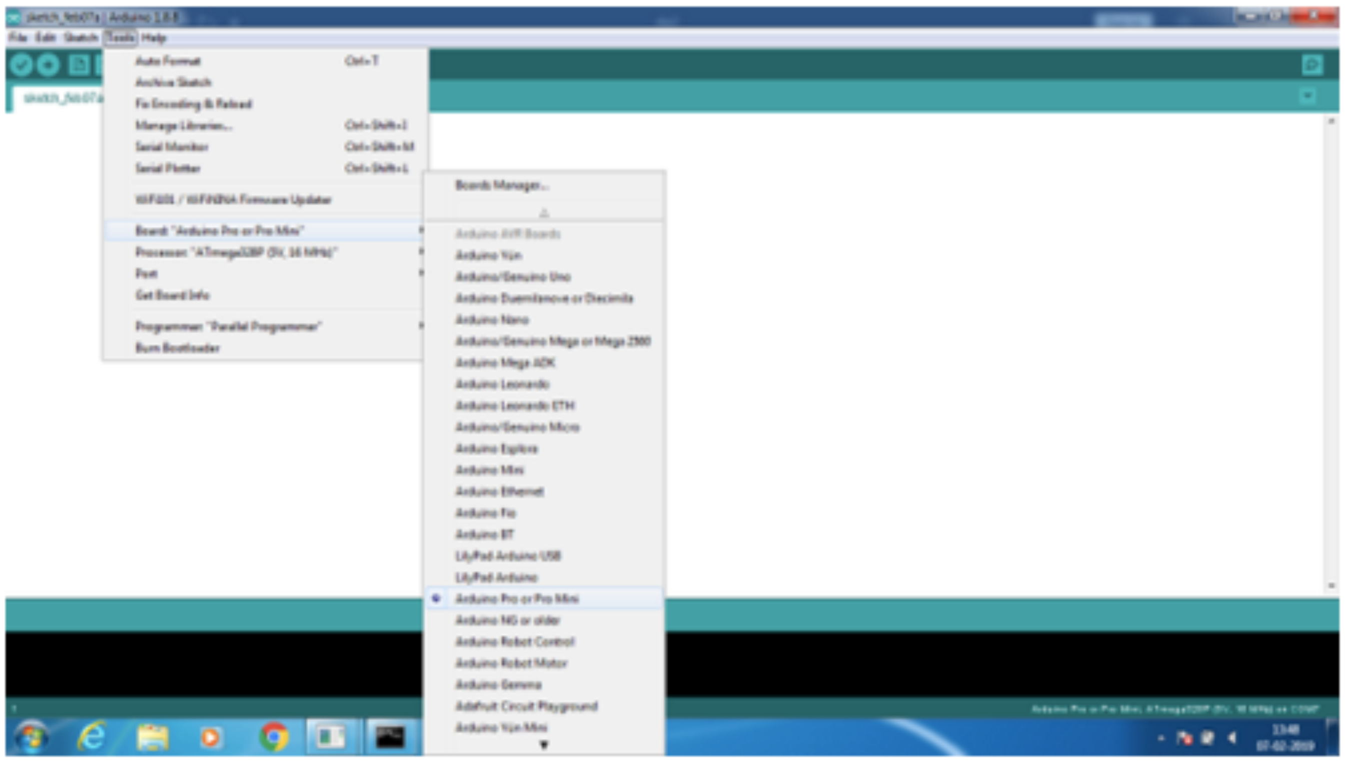


Fig 8. Removing Arduino I.C

Now connect Arduino Pro Mini as follow:-

Now go to Tools and select Board Arduino pro mini and upload the code to Arduino pro mini refer Fig 9.

Arduino Uno	Arduino Pro Mini
Rx	Rx1
Tx	Tx0
Gnd	Gnd
5V	VCC
Reset	Reset



Board: "Arduino Pro or Pro Mini"
Processor: "ATmega328P (5V, 16 MHz)"
Programmer: "Parallel Programmer"

Fig 9. Setting board

Now we need to connect the components

Arduino Pro Mini Pins	Components Pins
5V	VCC of Bluetooth
GND	GND of Bluetooth
RXI	TX of Bluetooth
TX0	RX of Bluetooth
VCC	OLED VCC
GND	OLED GND
A4(SDA)	OLED SDA
A5(SCL)	OLED SCL

Refer Fig. 10 for connections

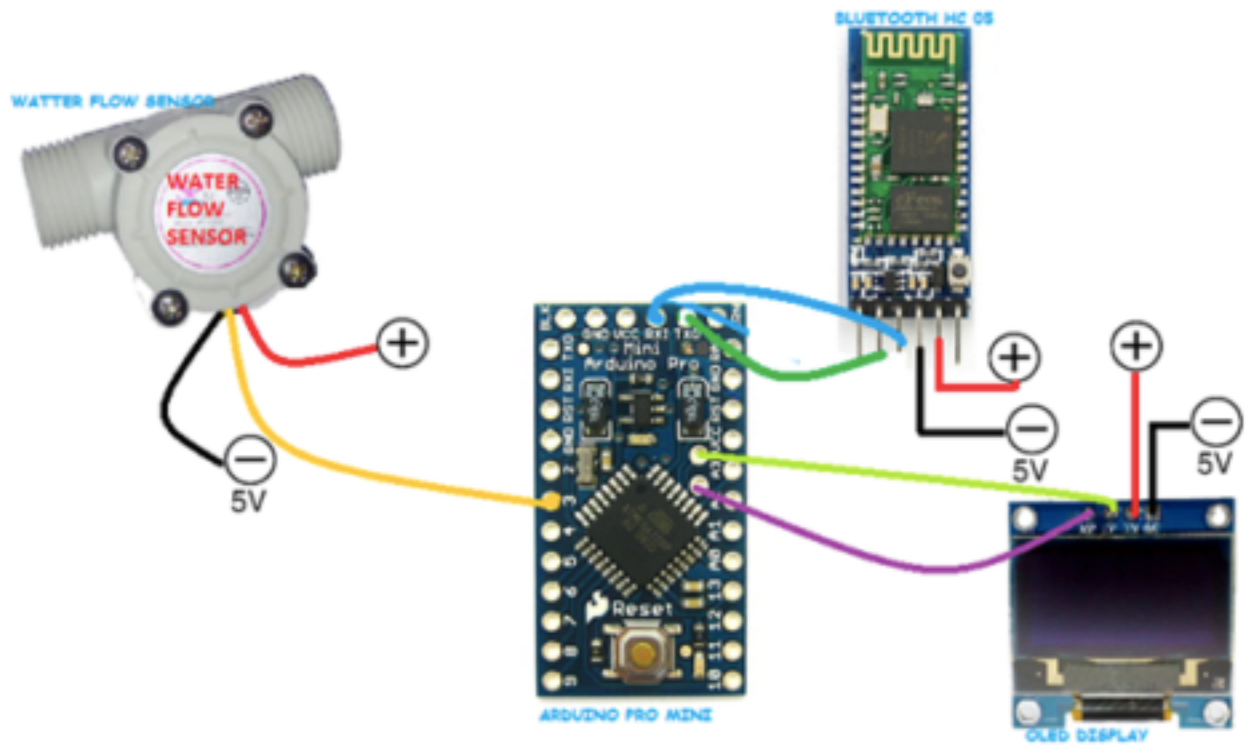


Fig 10. Connections

Do cross check all the connections, if you don't want to end up frying your Arduino with wrong connection.

First make sure each of the connections are ok and then power the Arduino with 5V DC power supply.

Connect the water sensor pipe to the tap or water inlet of your water tank. By doing so, you can see the total volume of water and water flow rate on OLED display. Now, open Android app and press the Bluetooth icon where you will see a list of Bluetooth devices available. Pair with Bluetooth HC 05. After connecting you will get the water flow rate and water volume consumed in the app as in Fig 11.

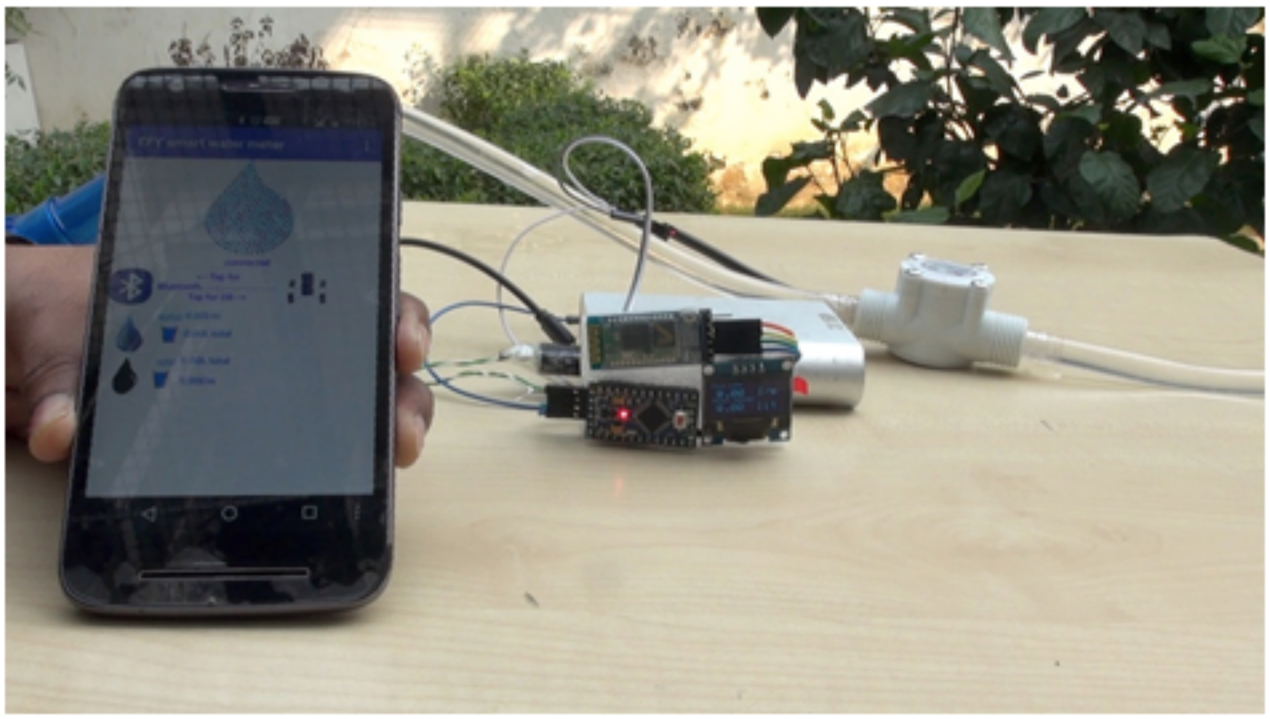


Fig 11. Working model

You can tap on DB icon to upload our water usage data to the web server that can be seen from anywhere using the internet.

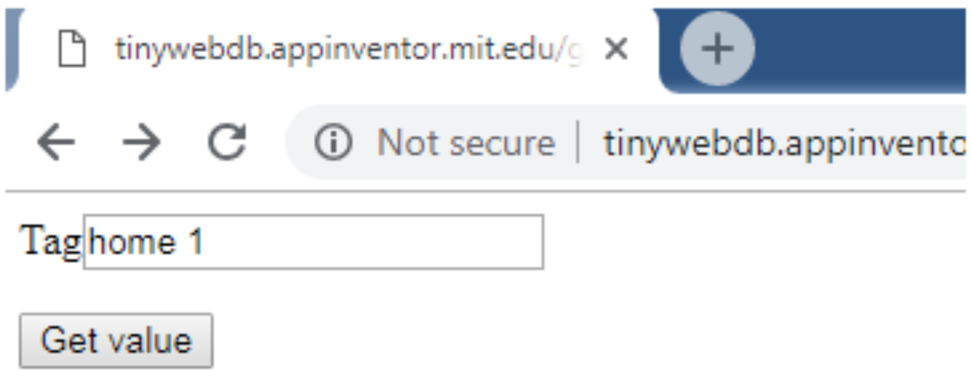
NOTE: Your internet connectivity must be turned on to upload data to the server.

To fetch data, click on the link given below and type your house name, that you have given in the app. I have used home 1. Click on search, You will get the details of water usage data. Refer Fig. 12.

Link for DB

Cheers! We have made an amazing solution that can help manage water consumption at home.

NOTE: You can also add the extra sensor to pin 2 for water wastage data. I have included code and app compatible with that. Only add a sensor to it and enjoy your project.



Tag home 1

Fig 12. Getting data from Web DB

Troubleshooting:-

Prob. (Problem): I don't find the accurate data from the sensor.

Sol. (Solution): Follow the library link for calibration tutorial.

Prob:- I get the "too large index", "end app" during running.

Sol:- This happens because some data might be missed by Bluetooth during reading, tap out on screen or reconnect the Bluetooth of the phone.

Prob:- Can I set it up and supply power through a socket?

Sol:- Use adaptor with 5V D.C for it.

Prob:- I can't find enough VCC pin to connect OLED, Bluetooth and sensor and as well as to power Arduino.

Sol:- Solder the extra wire to VCC and ground pins and make connections.

Prob:- I can't see Bluetooth HC 05 in the list.

Sol:- First pair the Bluetooth HC 05 with android, go on Bluetooth setting and tap on HC 05 to pair, and it will ask for the pairing code, default code for paring is 1234 or might be 0000 or 1111.

Prob:- Error in Bluetooth communication and program uploading to Arduino.

Sol:- Try interchanging the connections between Rx and Tx.

You can watch Video for this DIY here: https://youtu.be/-5OD6kmP3Gg

17

FINGERPRINT DOOR UNLOCK SYSTEM

BY DINU D. AND CINLA K. PAPPACHAN

This simple fingerprint door unlock project using Arduino can be very useful for door security, forensics, crime investigation, personal identification, attendance system and much more. In the future, there could be many more applications like fingerprint based driving licences, bank accounts operation and so on.

The whole system works under a simple algorithm called matching algorithm, which is used to compare previously-stored templates of fingerprints against users' fingerprints for authentication purposes.

A key is normally used for traditional door opening, but it provides very poor security. In this fingerprint door unlock project, only when an authorized person places a finger on the sensor, the door unlocks and the LCD displays a welcome message along with that person's name.

Circuit and working

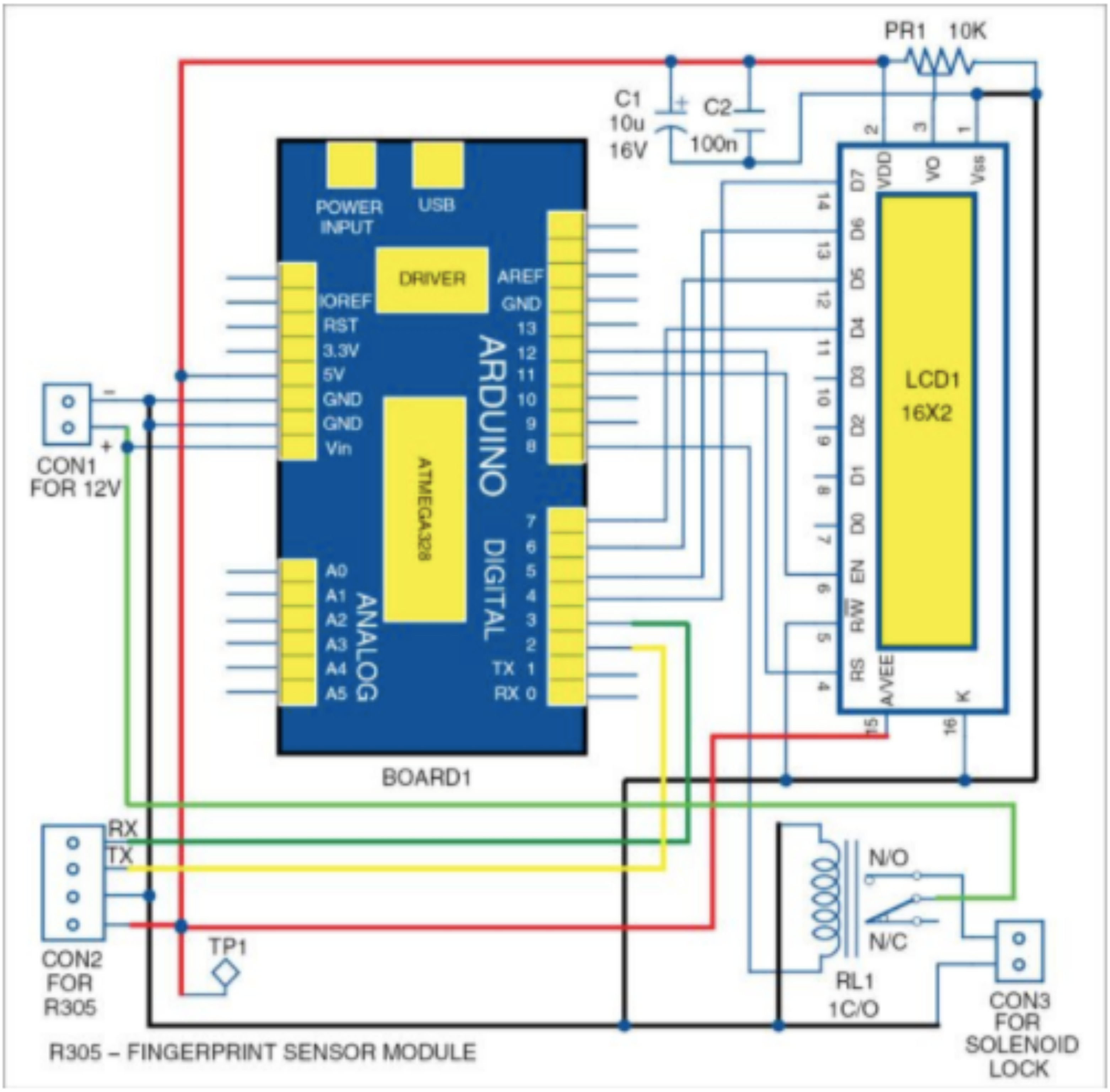


Fig. 1: Circuit diagram of the fingerprint door unlock system

The circuit shown in Fig. 1 operates using a 12V power supply. An Arduino microcontroller (MCU) requires only 5V but the solenoid electric lock requires 12V. As Arduino Uno has an inbuilt 5V voltage regulator, a common 12V supply can be used for the whole system.

Test Points

Test point	Details
TP1	5V

(Voltage measure w.r.t. GND)

PARTS LIST

Semiconductors:	
BOARD1	- Arduino Uno
Resistors (all 1/4-watt, ±5% carbon):	
PR1	- 10-kilo-ohm preset
Capacitors:	
C1	- 10µF, 16V electrolytic
C2	- 100nF, ceramic disk
Miscellaneous:	
RL1	- 5V, 1C/O relay
LCD1	- 16×2 alphanumeric display
CON1	- 2-pin connector
	- DC supply
CON2	- 4-pin connector
	- R305
CON3	- 2-pin connector
	- Solenoid lock

The brain of the circuit is Arduino Uno MCU board (BOARD1). It is based on ATmega328/ATmega328P and has 14 digital input/output (I/O) pins, six analogue inputs, 32k flash memory, 16MHz crystal oscillator, a USB connection, power jack, ICSP header and reset button, among others. It can be programmed using Arduino IDE software.

Fingerprint sensor module R305 (connected across CON2) has UART interface with direct connections to the MCU or to the PC through max232/USB serial adaptor. The user can store fingerprint data in the module and configure it in 1:1 or 1:N mode for identification. Pins TX and RX of R305 sensor are connected to Arduino digital pins 2 and 3, which are used for serial communication.

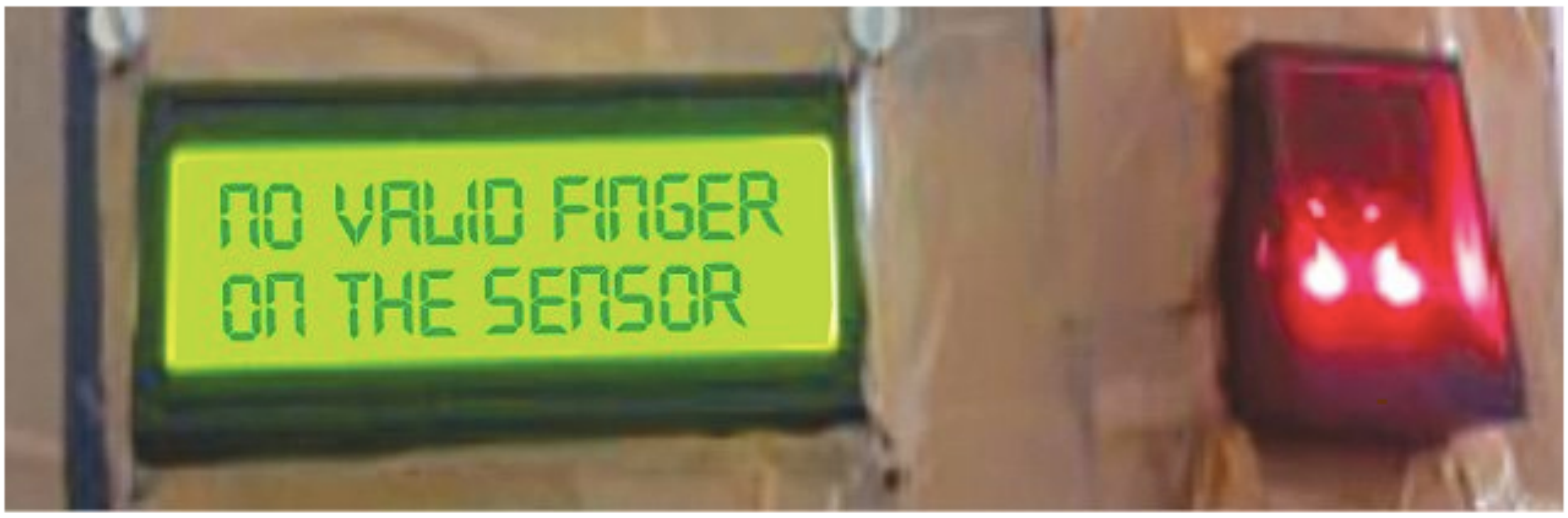


Fig. 2: Initial state

The LCD display (LCD1) is used to display messages during action. Here, a 16x2 display is used; each character is made of 5x7 dot-matrix. Pins 3, 4, 5 and 6 of the LCD are the control lines connected to preset (PR1) output, pin 12 (Arduino), GND and pin 11 (Arduino). Pins 11, 12, 13 and 14 are data pins of the LCD that are connected to pins 7, 6, 5 and 4 of Arduino, respectively. Preset PR1 is used to adjust the contrast of the LCD display.

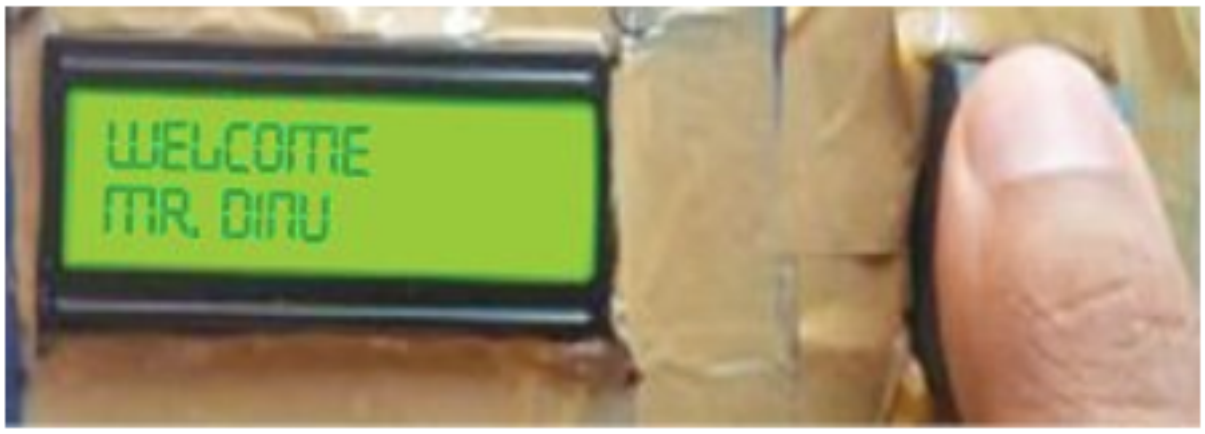


Fig. 3: Valid finger

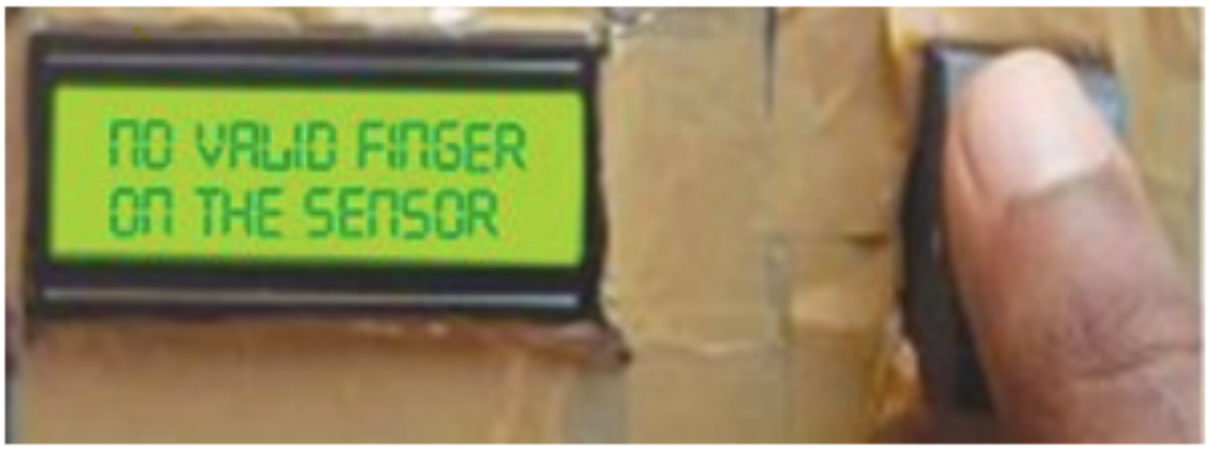


Fig. 4: Invalid finger

An electronic door-lock solenoid (connected across connector CON3) is basically an electromagnet made of a big coil of copper wire with an armature (slug of metal) in the middle. When the coil is energised, the slug is pulled into the centre of the coil. This allows the solenoid to move to one end.

The solenoid lock requires more current than what Arduino can provide. Therefore to operate the lock, a 5V relay (RL1) is used. CON3 is connected between normally open (N/O) contacts of RL1 and GND. The sequence of messages on the LCD from author's prototype are shown in Figs 2, 3 and 4.

Software

Programs named enroll and fingerprint use different functions like getFingerprintEnroll(int,id), Adafruit_Fingerprint(&mySerial) and getFingerprintEnroll(id). These functions are defined inside the library and pass arguments when called.

After uploading enroll in the Arduino, open serial monitor from Arduino IDE from Tools→Serial monitor options. Change baud rate below the serial monitor window to 38400. Choose Newline option from the same place. Then, follow the instructions on the serial monitor. Place the finger on the fingerprint module. Type any whole number as the ID number. Press Send tab to send the ID number from the serial monitor to Arduino. This fingerprint gets converted into digital data and gets store inside R305 module database.

More than 200 fingerprints can be stored on this system. Make sure that each fingerprint has a unique ID number. This ID number will be used in the next program to identify the authenticated person's name. The serial monitor will guide the user as to when he or she should place the finger and when to remove it.

For debugging without an LCD display, make the same settings for the serial monitor after uploading Fingerprint program. This is used to compare the fingerprint in the sensor with stored prints. The serial monitor guides here also. The fingerprint program should be edited to change the name and ID numbers according to how users want.

Download source code: click here

Construction and testing

The single-side PCB for the fingerprint door unlock system is shown in Fig. 5 and it's component layout in Fig. 6. For convenience, we have designed the PCB as an Arduino shield. The users can modify the design as per requirement. Also, they can test PCB with Arduino board using a cable connector.

Download PCB and component layout PDFs: click here

Make sure the baud rate given in the program is correct. Baud rate for the serial monitor can be anything but the baud rate for R305 sensor should match that given in its datasheet. Baud rate may vary with different versions of the sensor. It is given in the program like Serial.begin(38400) [baud rate for serial monitor]; finger.begin(57600) [baud rate for sensor]. Reset Arduino board before validation of the fingerprint.

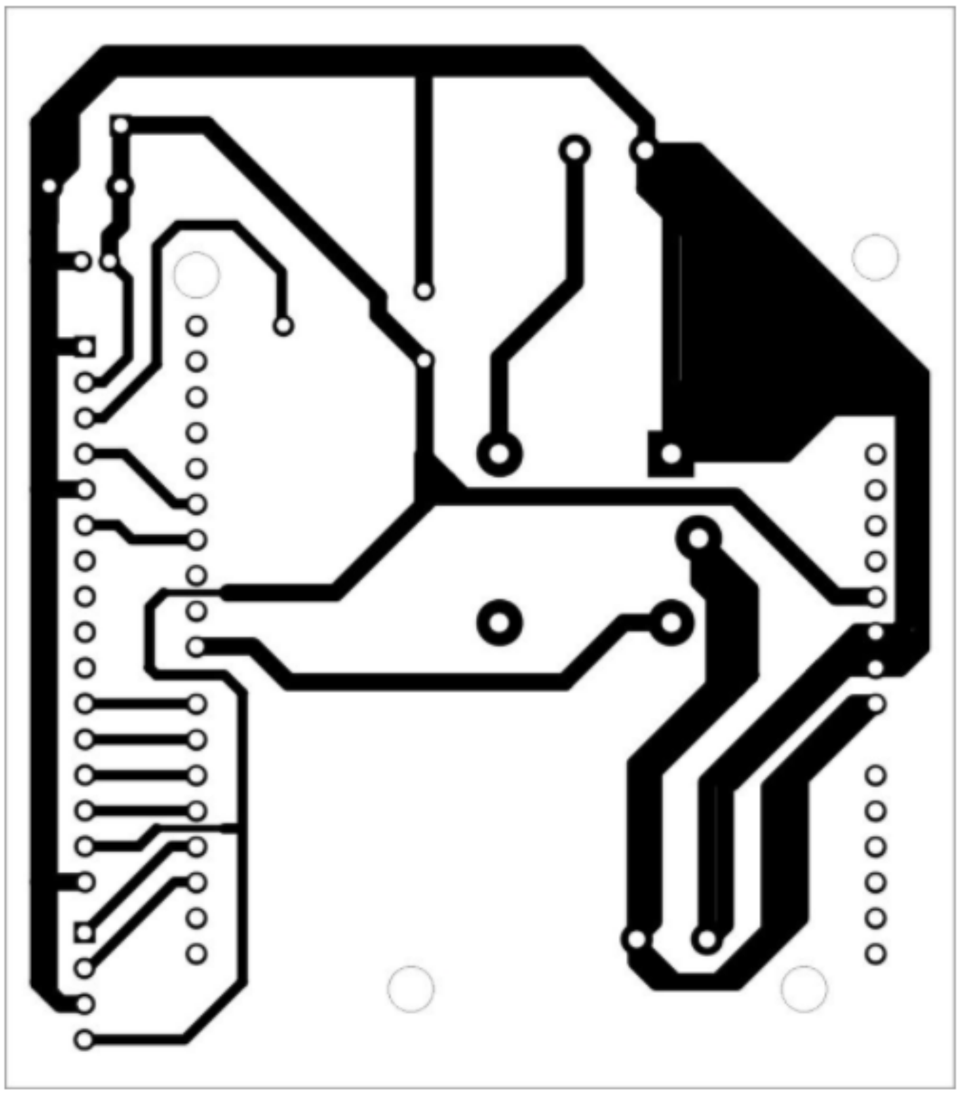


Fig. 5: PCB pattern of the fingerprint door unlock system

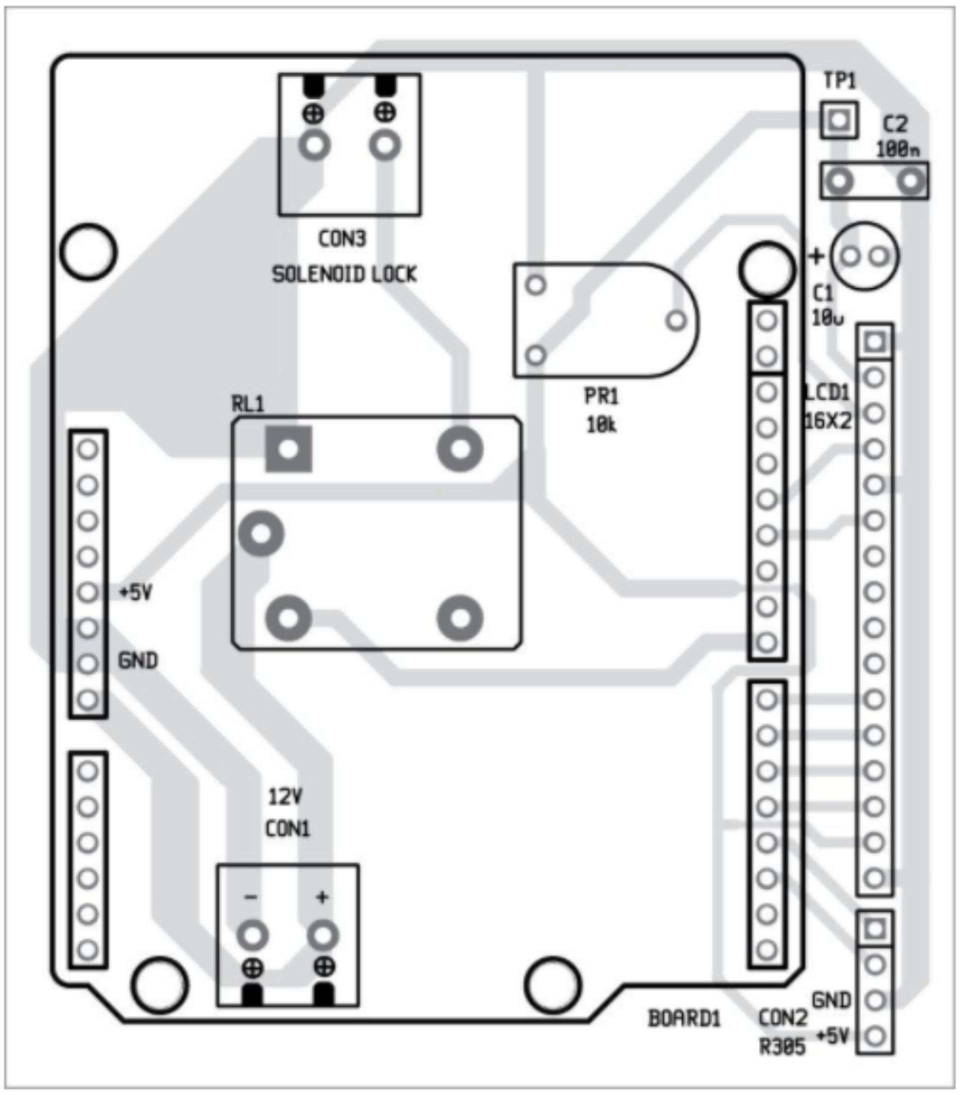


Fig. 6: Component layout of the PCB

18

LM386 AUDIO AMPLIFIER

BY RAJ K. GORKHALI

Here is a simple LM386 based audio amplifier circuit with the author's prototype shown below.

LM386 based audio amplifier: circuit and working

Circuit diagram of the LM386 based audio amplifier is shown in Fig. 2. It is built around popular amplifier LM386 (IC1), an 8-ohm, one-watt speaker (LS1), four capacitors and a few other components. A 6V battery is used to power this project.

Four electrolytic capacitors [two 10μF, 16V (C1 and C2) and two 220μF, 16V (C3 and C4)] are used in this circuit. C1 is connected to the middle terminal of 10k potmeter VR1. C2 is connected to pins 1 and 8 of IC1. Pin 5 of IC1 is its output terminal, which is connected to speaker LS1 through C3.

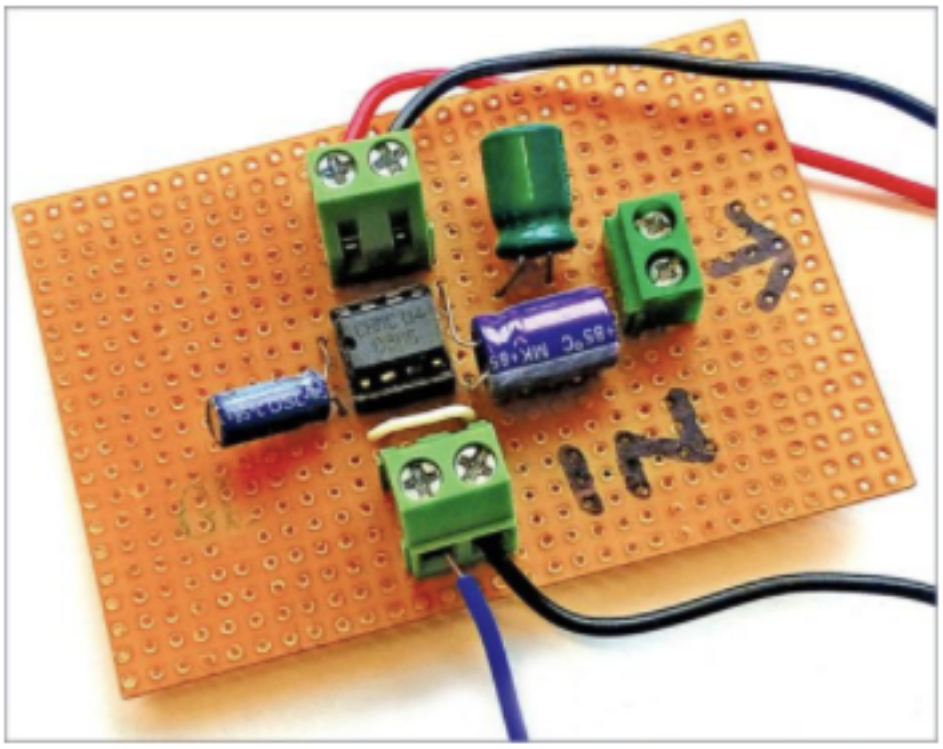


Fig. 1: Author's prototype of LM386 based audio amplifier

C4 is connected to the positive terminal of 6V battery and ground. Positive side of 6V is connected to pin 6 of IC1 and the other side to ground terminal to pin 4.

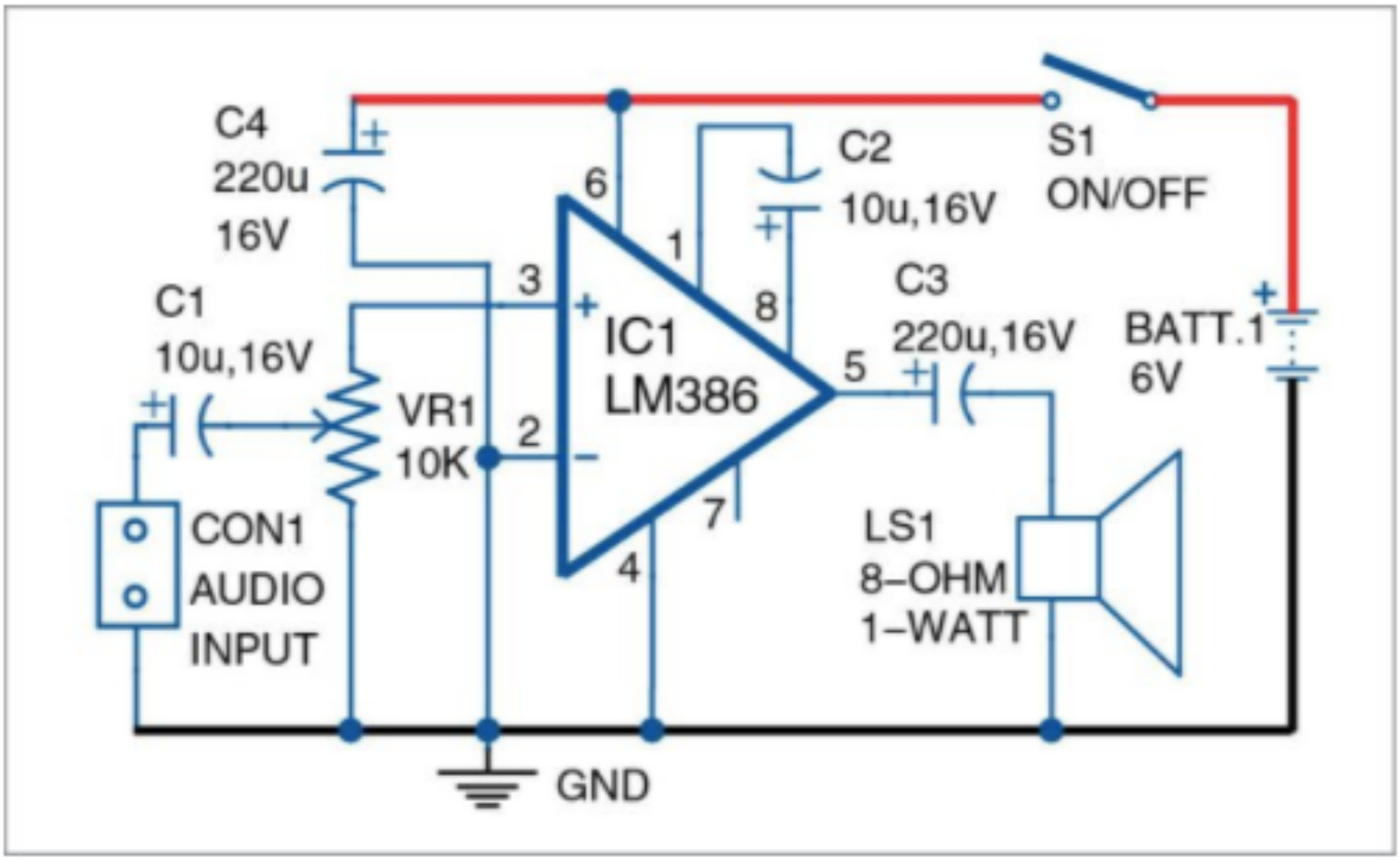


Fig. 2: Circuit diagram of the LM386 based audio amplifier

Inverting pin 2 of IC1 is connected to ground and non-inverting pin 3 is connected to the input terminal through VR1. Audio input is fed to CON1. VR1 is used to control volume.

Construction and testing

A single-side PCB for LM386 amplifier is shown in Fig. 3 and its component layout in Fig. 4. After assembling the circuit on a PCB, enclose it in a suitable box. Fix connector CON1 on the front panel for input and loudspeaker LS1 at the rear side of the box. Connect VR1 on the front panel for controlling the volume.

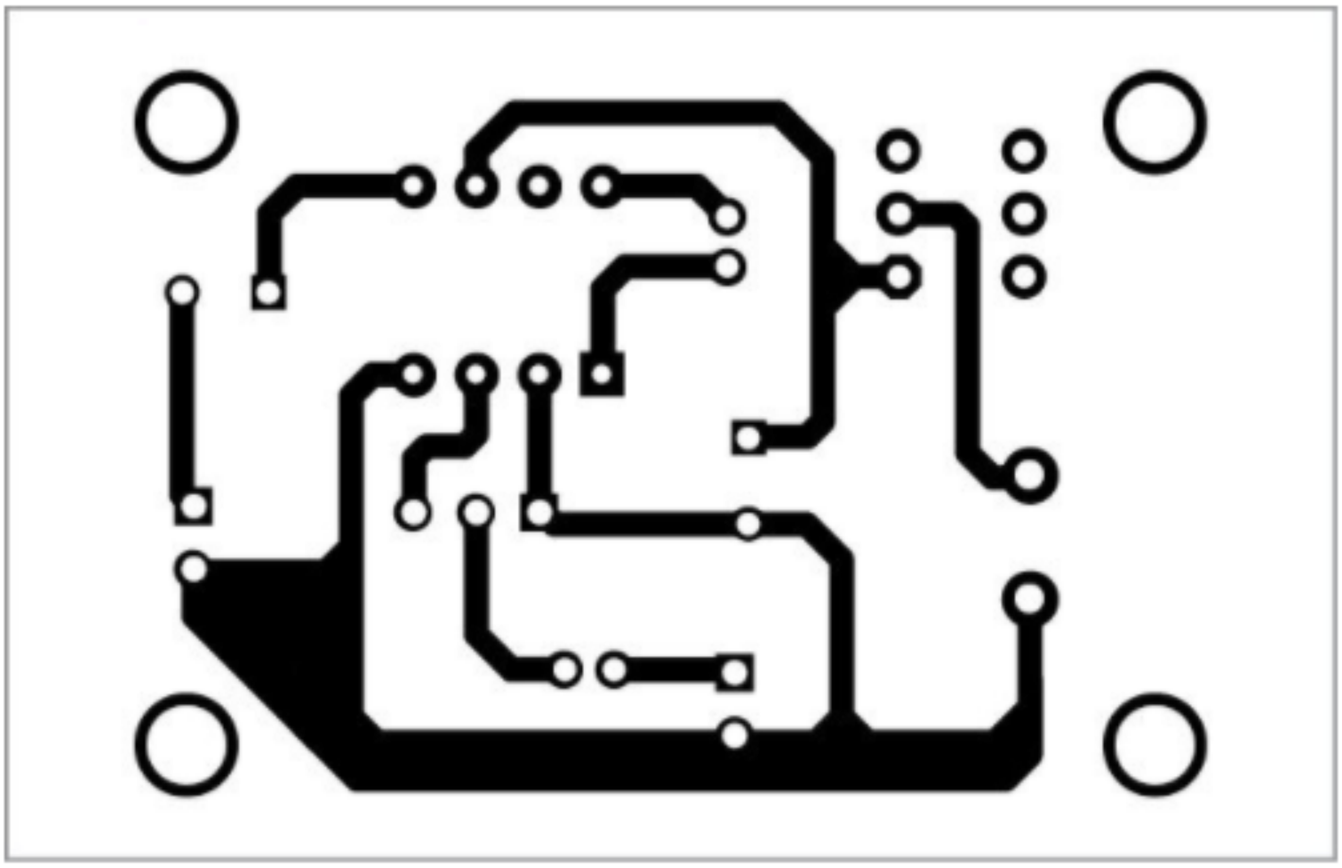


Fig. 3: PCB pattern of the LM386 based Audio Amplifier

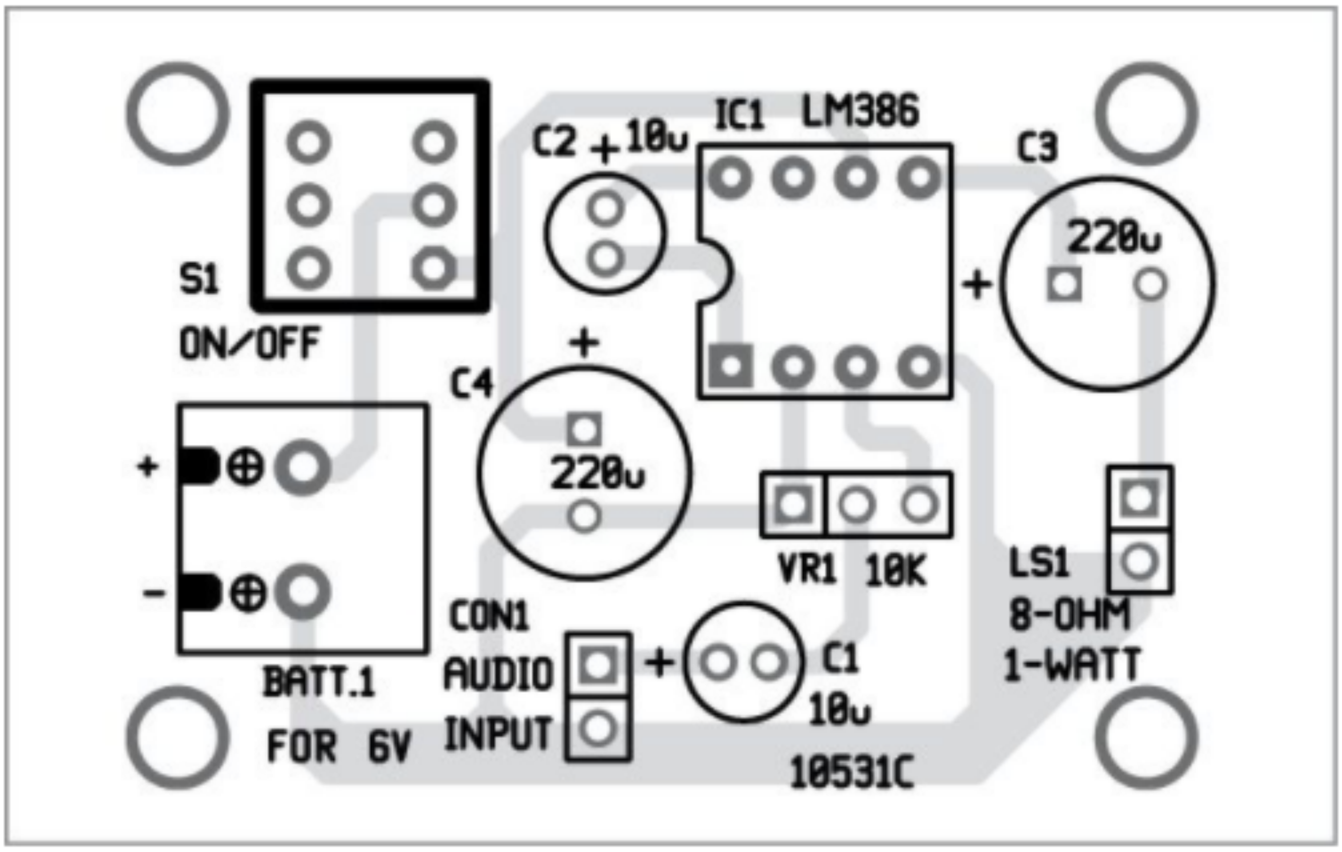


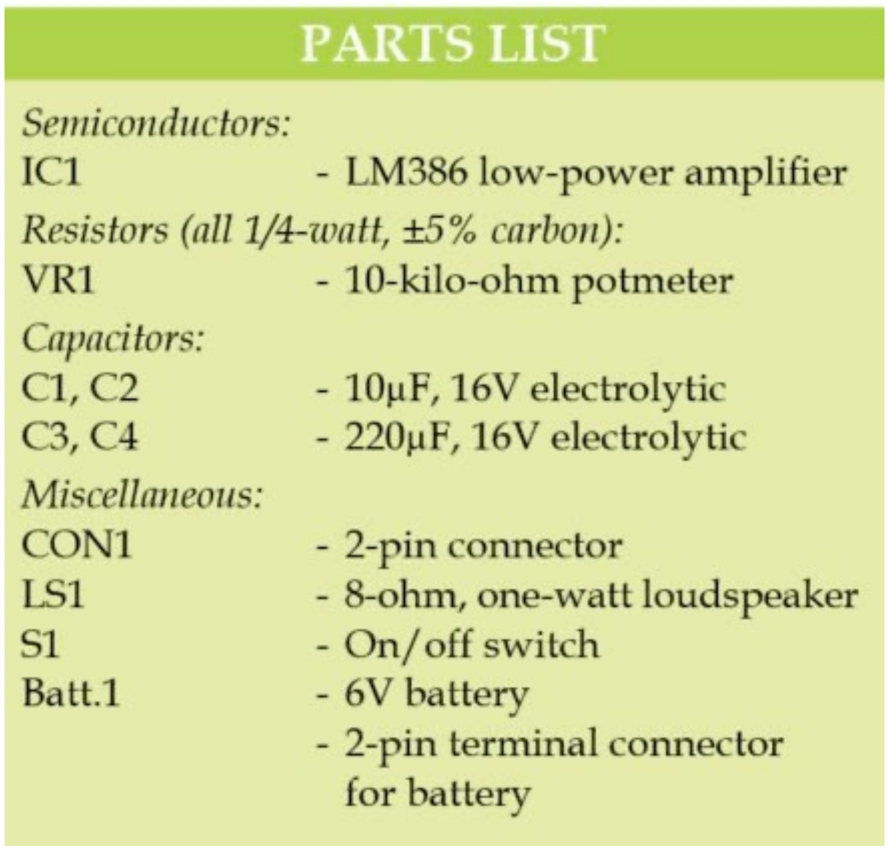
Fig. 4: Component layout of the PCB

Before using this project, test it using the 6V battery. Connect the 8-ohm, one-watt speaker to output pin 5 of IC1 through C3.

Download PCB and component layout PDFs: click here

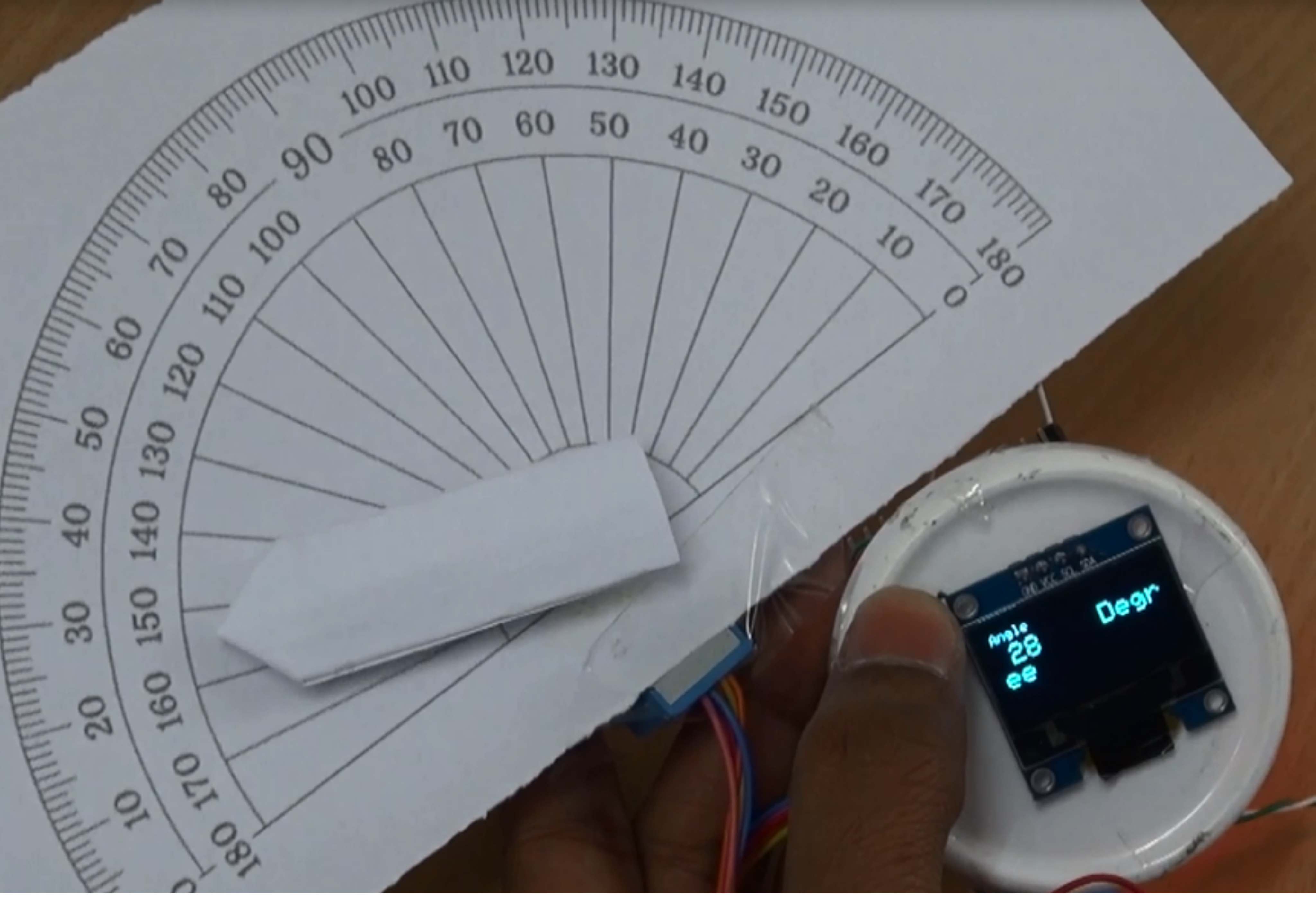
Switch on S1 and keep VR1 to its mid position. Now, take a screwdriver and gently touch it on input terminal pin 3 of IC1. You should hear a humming sound from the speaker. This will confirm that your circuit is working and ready to use.

Note. LM386 provides an output of 250 milliwatts to one watt depending on supply voltage and load. Refer its datasheet for details.



PARTS LIST

Semiconductors:	
IC1	- LM386 low-power amplifier
Resistors (all 1/4-watt, ±5% carbon):	
VR1	- 10-kilo-ohm potmeter
Capacitors:	
C1, C2	- 10µF, 16V electrolytic
C3, C4	- 220µF, 16V electrolytic
Miscellaneous:	
CON1	- 2-pin connector
LS1	- 8-ohm, one-watt loudspeaker
S1	- On/off switch
Batt.1	- 6V battery
	- 2-pin terminal connector for battery



19

DIGITAL PROTRACTOR AND ANGLE MEASUREMENT DEVICE WITH ARDUINO

BY ASHWINI KUMAR SINHA

A compass and a protractor are two of the most basic tools used in geometry. For mathematics and engineering students, these tools are a must. But sometimes it is difficult to get accurate angle measurement for certain structures and geometrical shapes using these traditional tools.

So, I thought of developing a digital compass to make the work a bit easier. Here I will demonstrate the making of a DIY prototype.

When this idea struck my mind, I came up with two different methods to make the digital compass - first converting the MPU6050 gyro sensor data into angle, and second by converting stepper motor rotation into angle. Now, I'm going to show you both the methods here.

First method -

Digital Compass Using Stepper Motor:

For this DIY project, we need the following components:

Components:

- Arduino Pro Mini
- ULN20003A Module
- 28BYJ Stepper Motor
- 2 Push Button
- OLED Display (I2C Interface)
- 5V Battery
- Wires

Coding:

First, we will set up libraries to Arduino IDE. For this, go to library manager and search "MPU6050 Tocken" and install the library. After successful installation, search for Stepper Motor and then install the "28BYJ stepper motor "library as in (Fig. 1 and 2,3).

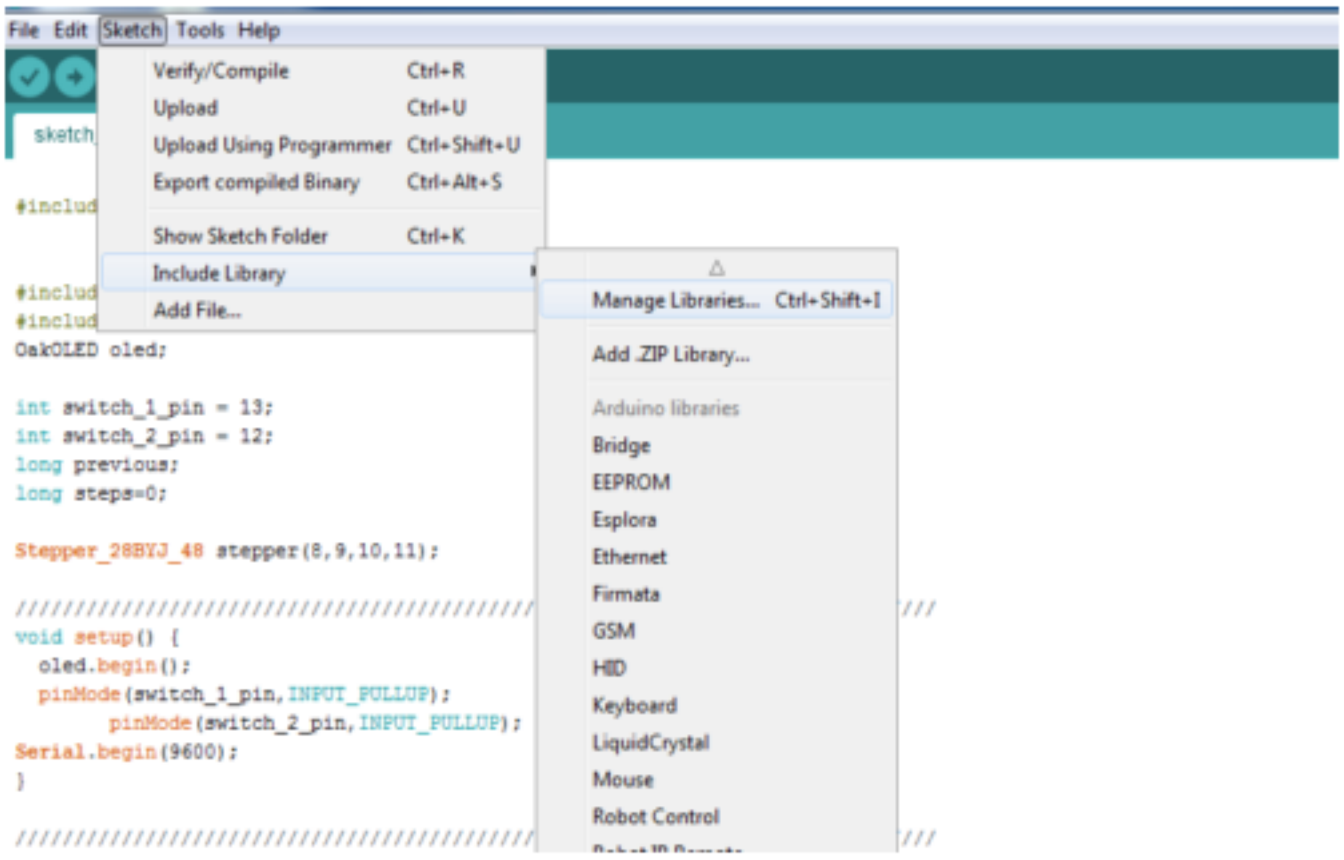
You can download the whole code from the link given at the end of this page.

As the required library has been installed now, we can start coding.

First, we will add the required library and declare the variables needed to store different values according to code snippet below (Fig. 4)

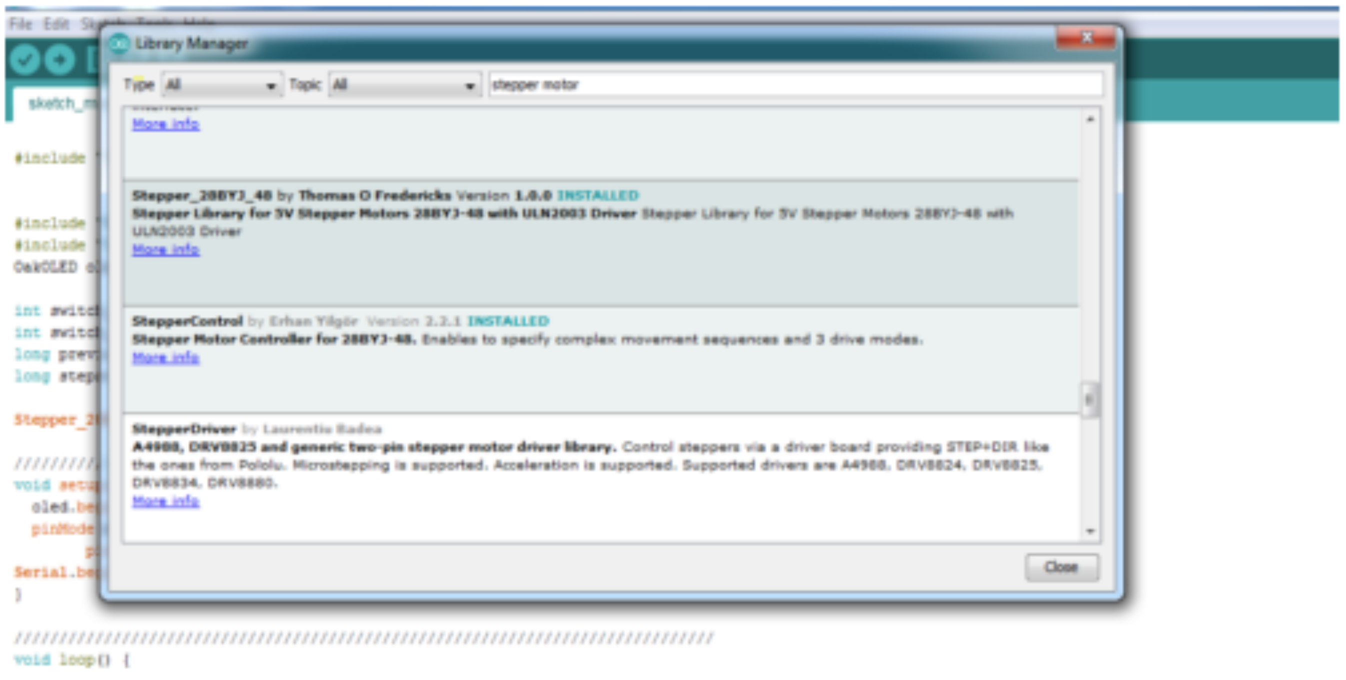
After this we will create a setup () function as described in Fig.5.

Next, we will create a loop function, where we will include 'if () condition' to check the state of the push button. When if () condition in code says true, stepper



```
#includ
#includ
#includ
OakOLED oled;

int switch_1_pin = 13;
int switch_2_pin = 12;
long previous;
long steps=0;

Stepper_28BYJ_48 stepper(8,9,10,11);

/////////////////////////////////////////////
void setup() {
  oled.begin();
  pinMode(switch_1_pin,INPUT_PULLUP);
        pinMode(switch_2_pin,INPUT_PULLUP);
Serial.begin(9600);
}

/////////////////////////////////////////////
```

Fig 1. Arduino opening Library manager



Stepper_28BYJ_48 by Thomas O Fredericks Version 1.0.0 INSTALLED
Stepper Library for 5V Stepper Motors 28BYJ-48 with ULN2003 Driver Stepper Library for 5V Stepper Motors 28BYJ-48 with ULN2003 Driver
More info

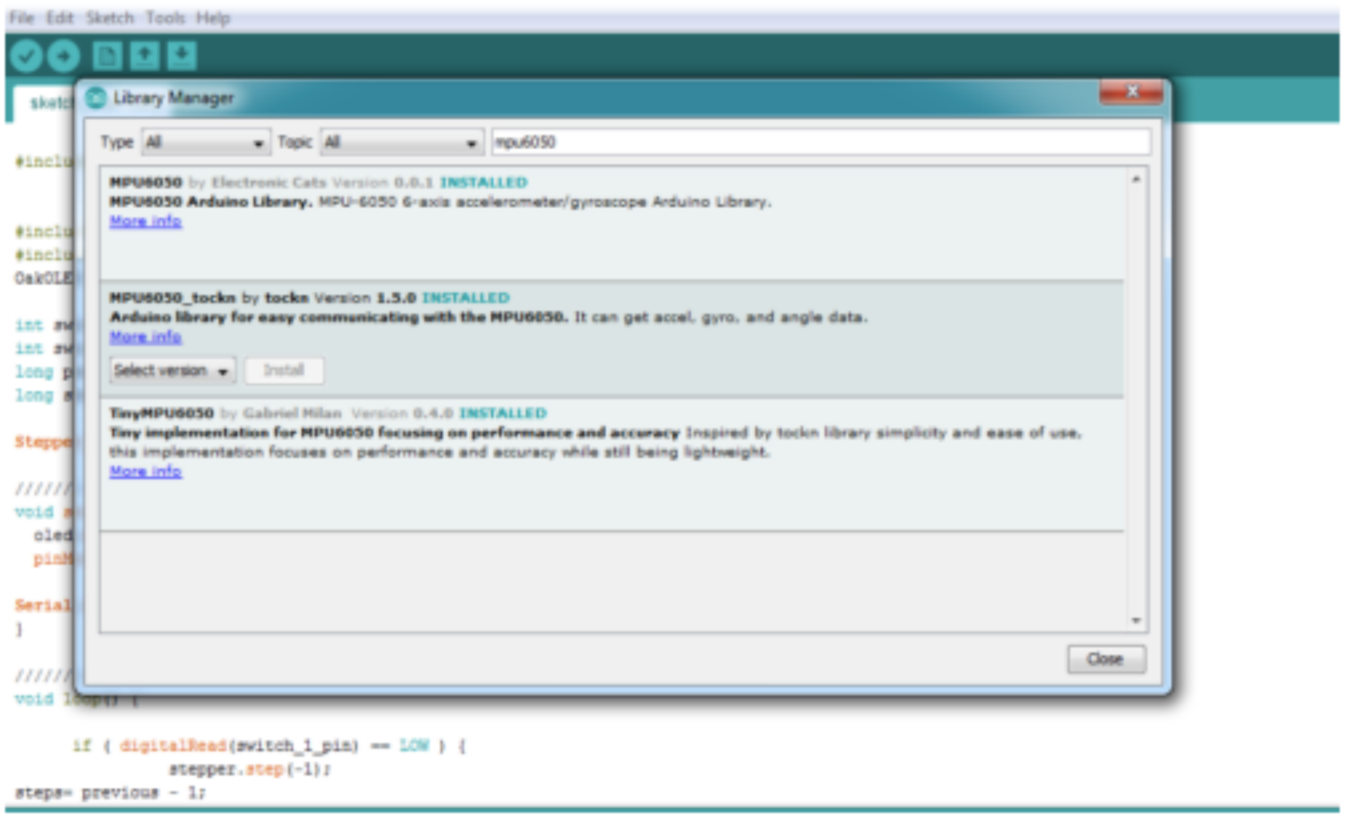
StepperControl by Erhan Yilgör Version 2.2.1 INSTALLED
Stepper Motor Controller for 28BYJ-48. Enables to specify complex movement sequences and 3 drive modes.
More info

StepperDriver by Laurentiu Badea
A4988, DRV8825 and generic two-pin stepper motor driver library. Control steppers via a driver board providing STEP+DIR like the ones from Pololu. Microstepping is supported. Acceleration is supported. Supported drivers are A4988, DRV8824, DRV8825, DRV8834, DRV8880.
More info

```
//////////////////////////////////////////////////////////////////
void loop() {
```

Fig 2. Installing stepper motor library

motor moves one step and stays there until the condition turns false (which happens when the switch is not pressed).



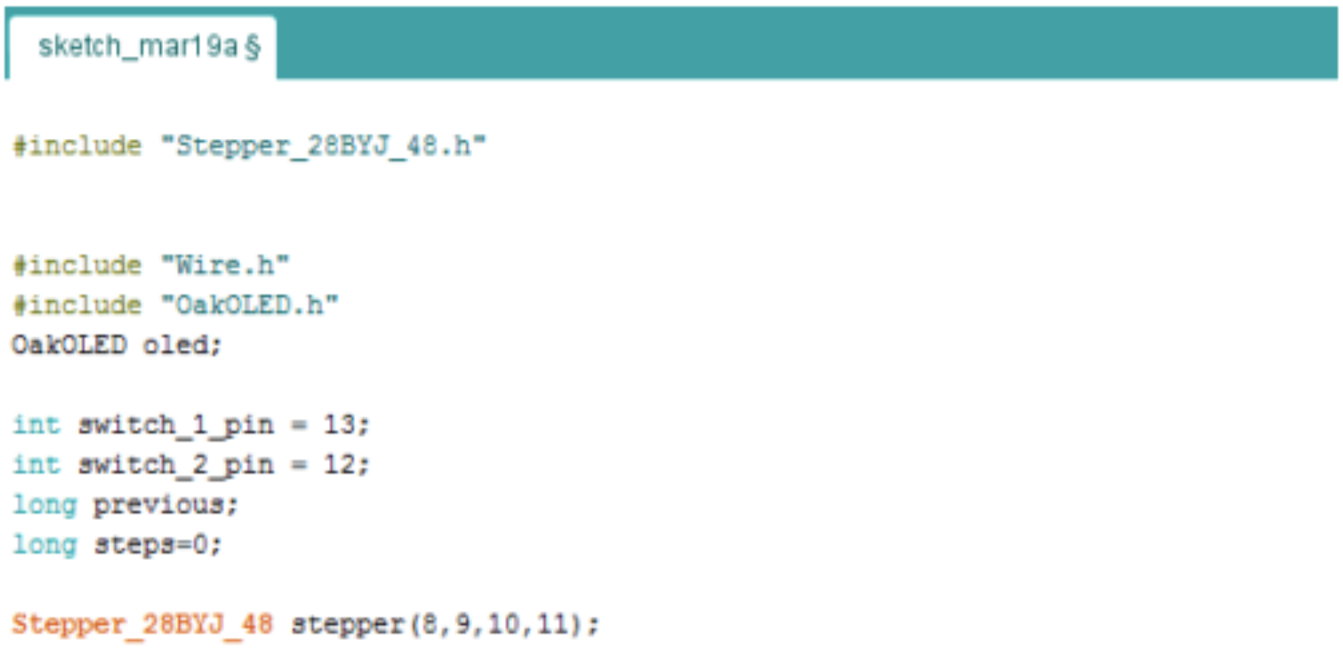
MPU6050 by Electronic Cats Version 0.0.1 INSTALLED
MPU6050 Arduino Library. MPU-6050 6-axis accelerometer/gyroscope Arduino Library.
More info

MPU6050_tockn by tockn Version 1.5.0 INSTALLED
Arduino library for easy communicating with the MPU6050. It can get accel, gyro, and angle data.
More info

TinyMPU6050 by Gabriel Milan Version 0.4.0 INSTALLED
Tiny implementation for MPU6050 focusing on performance and accuracy Inspired by tockn library simplicity and ease of use, this implementation focuses on performance and accuracy while still being lightweight.
More info

```
      if ( digitalRead(switch_1_pin) == LOW ) {
             stepper.step(-1);
steps= previous - 1;
```

Fig 3. Installing Mpu6050 Library



```
#include "Stepper_28BYJ_48.h"


#include "Wire.h"
#include "OakOLED.h"
OakOLED oled;

int switch_1_pin = 13;
int switch_2_pin = 12;
long previous;
long steps=0;

Stepper_28BYJ_48 stepper(8,9,10,11);
```

Fig 4. Arduino Code

To measure the angle, we have to calculate the steps moved by the motor. To do so, I have created a step variable that will store the previous value of stepper motor and go on adding one by one as the motor moves. Are you still confused? Let me explain it in simple words.

Suppose the initial value of the step variable is zero. When the motor moves one step, then its value will change to 0+1. This value is assigned to another variable named “previous”, so the previous value is now 1 and when the motor moves another step the new value becomes previous value + 1. In this way, each time the motor moves a step, the number of steps automatically get added. For example

Step = 0+1
Previous = 0

Step = 1+1
Previous =1

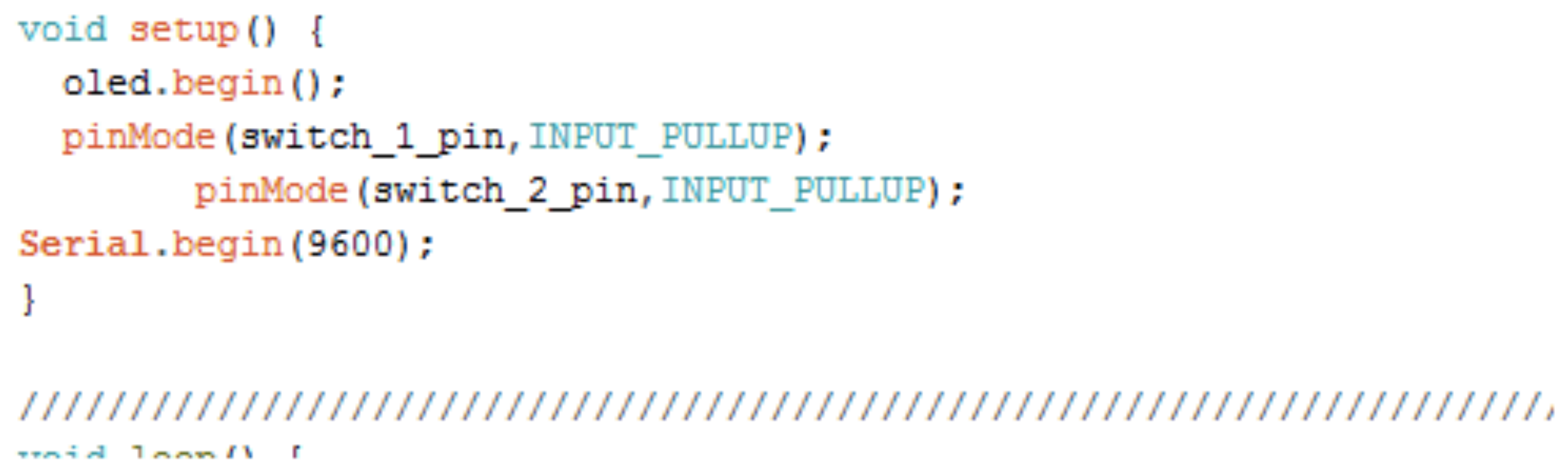
Step =2+1
Previous =2

This is how the steps of motor are calculated.

After this, we need to convert the steps of motor into angle by using a map.
Map (min steps, max step, min angle, max angle)

The magic happens here! When we put this above part of code in loop function, the steps automatically get converted into angle.

In order to map the value of stepper motor, we need to know the max number of steps it takes to complete one full rotation, this depend on the resolution of our stepper motor.



```
void setup() {
  oled.begin();
  pinMode(switch_1_pin,INPUT_PULLUP);
        pinMode(switch_2_pin,INPUT_PULLUP);
Serial.begin(9600);
}

//////////////////////////////////////////////////////////////////////
```

Fig 5. Setup function.

Here in this demonstration, my stepper motor takes nearly 510 steps to complete one rotation. (Refer to Fig .6)

Now we are done with the coding part. Let's connect our components.
Connection:

Connect all the components as stated below or Refer Fig .7 for connection

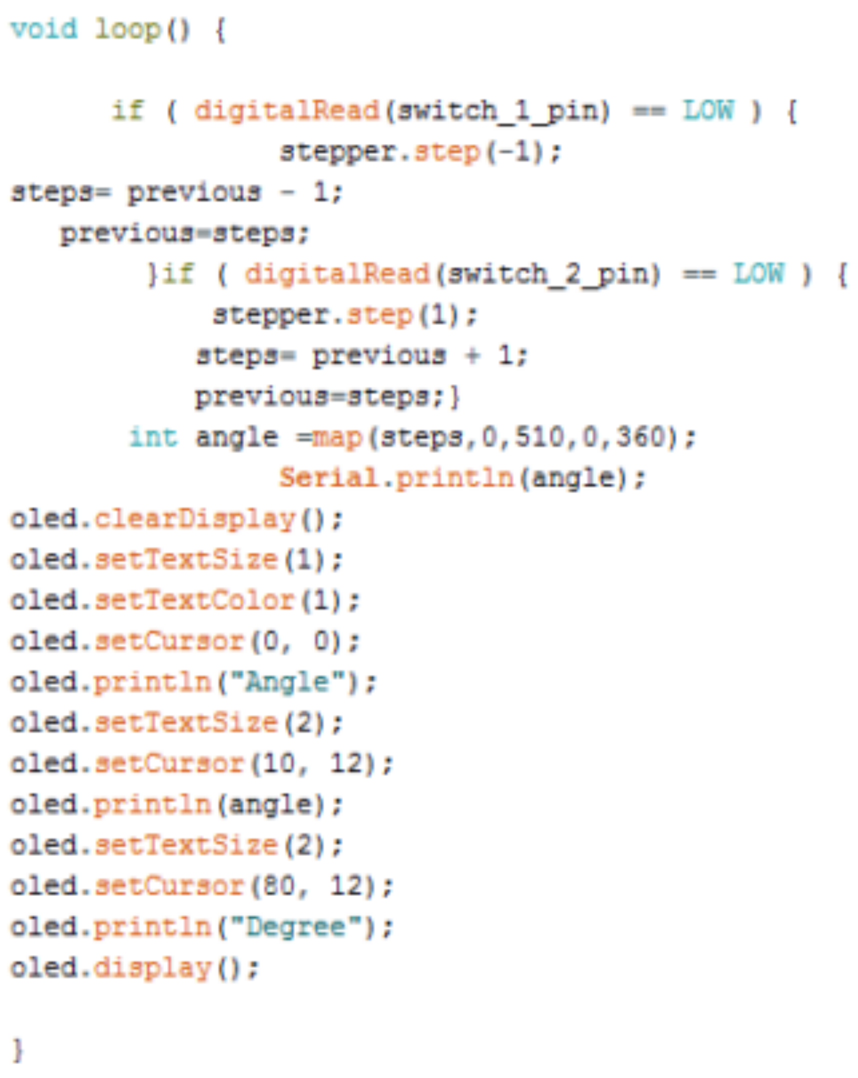
Arduino	Component
SCL	SCL OLED
SDA	SDA OLED
GND	GND OLED
VCC	VCC OLED
PIN 8	ULN IN 1
PIN9	ULN IN 2
PIN10	ULN IN 3
PIN 11	ULN IN 4
PIN 12	SWITCH 2 PIN 1
PIN 13	SWITCH 1 PIN 1
VCC	SWITCH PINS 2 (1, 2)

Testing

Crosscheck every connection, as any mistake might fry your costly components.

After checking all connections, power the Arduino and motor driver module. When we press the switch, the stepper motor arrow will start moving and at the same the OLED display will show the angle moved by the stepper motor. If you press the other switch, the stepper will start moving in other direction and the OLED will show the angle moved in that direction.

You can measure the angle of any structure with this digital tool. You



```
void loop() {

      if ( digitalRead(switch_1_pin) == LOW ) {
                stepper.step(-1);
steps= previous - 1;
   previous=steps;
        }if ( digitalRead(switch_2_pin) == LOW ) {
            stepper.step(1);
           steps= previous + 1;
           previous=steps;}
       int angle =map(steps,0,510,0,360);
                Serial.println(angle);
oled.clearDisplay();
oled.setTextSize(1);
oled.setTextColor(1);
oled.setCursor(0, 0);
oled.println("Angle");
oled.setTextSize(2);
oled.setCursor(10, 12);
oled.println(angle);
oled.setTextSize(2);
oled.setCursor(80, 12);
oled.println("Degree");
oled.display();

}
```

Fig 6. Arduino code calculating angle.

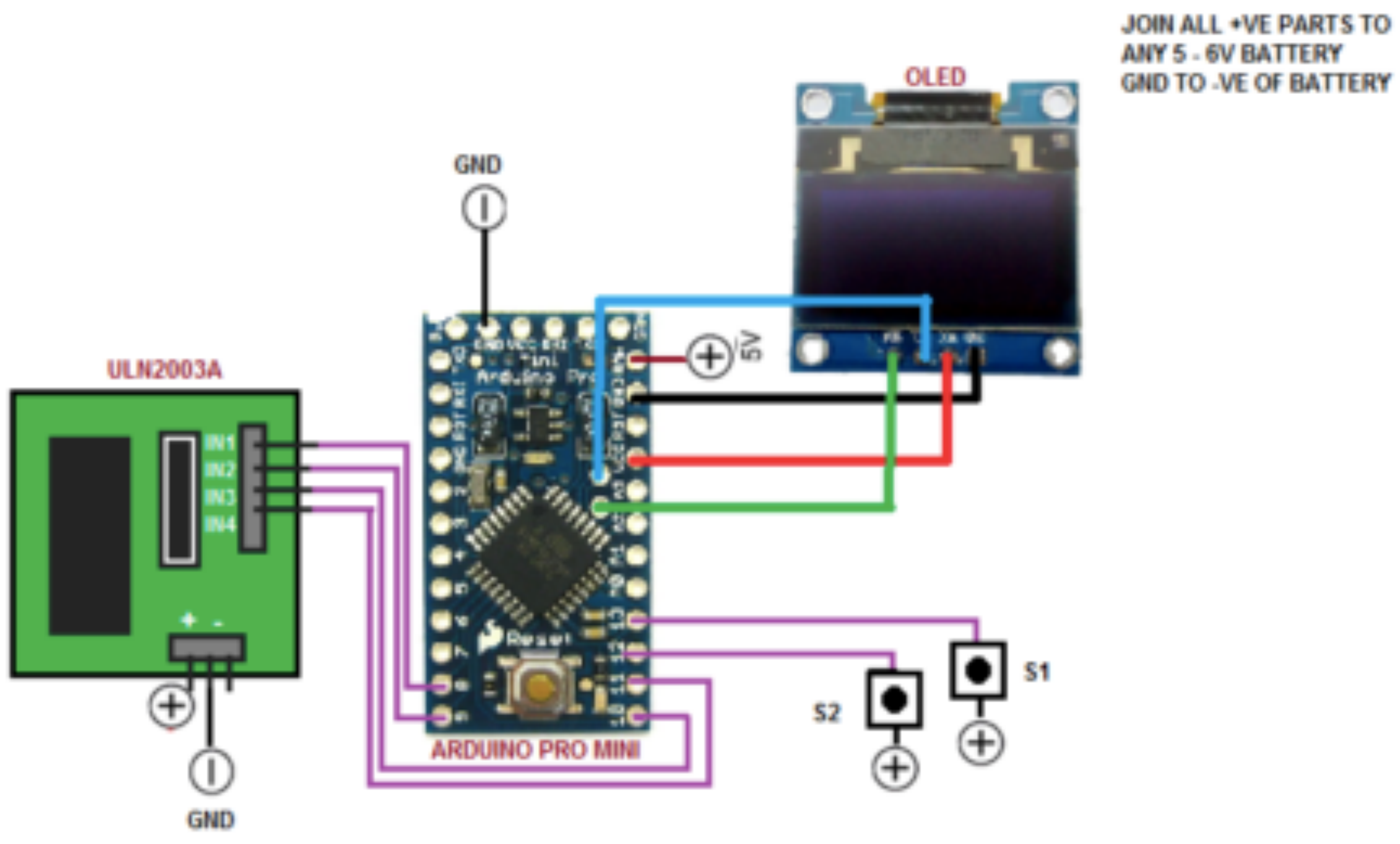


Fig .7 Connection

just press one switch and match the stepper arrow with reference to the surface of the structure and the OLED display will show you the angle measurement. You can also use it to draw arc and angle on any Engineering drawing sheets.

Digital angle measuring Compass with MPU 6050 gyro sensor

We just made a digital compass with stepper motor. How about making another one with MPU 6050? This one may be more interesting because it tells the angle of all 3 coordinates (X,Y,Z) at the same time.

To make the digital compass with MPU6050, we need to gather the following components.

Components

- Arduino pro mini
- MPU 6050
- OLED Display(I2C) Interface
- Battery 5V

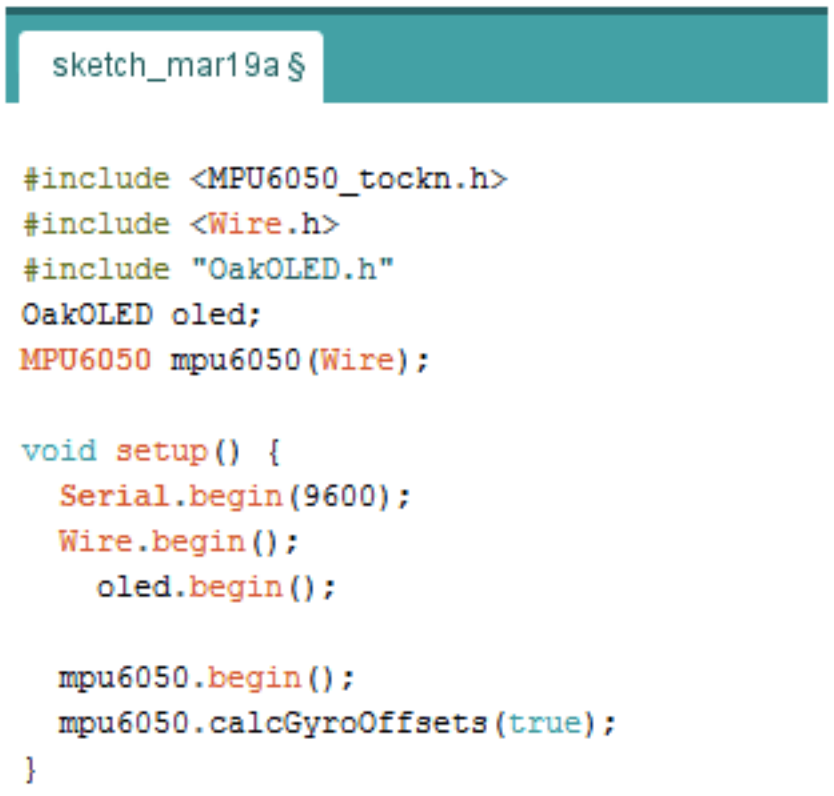
• Wires

Coding

As we have collected the components, we can now start coding.

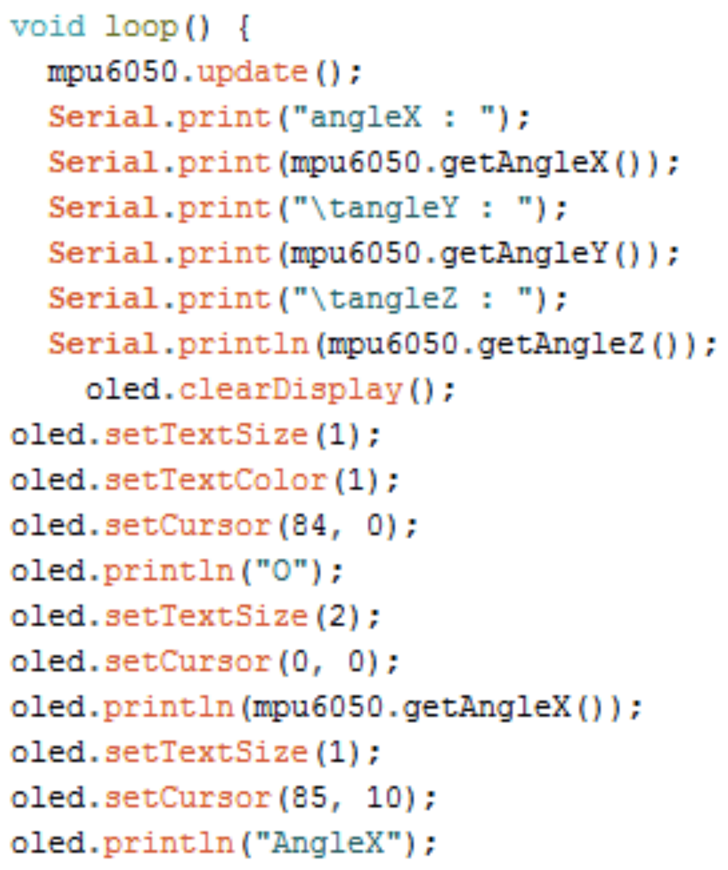
In the first part of coding, we will add MPU6050 and OLED Library and then create a setup function as in Fig 8.

After this, we will now create a loop function and try to get the angle movement of MPU 6050 using keyword “getAngleX” . You can make it display on the OLED screen, use keyword “oled.print(text to print)” (Refer Fig .9).



```
sketch_mar19a §

#include <MPU6050_tockn.h>
#include <Wire.h>
#include "OakOLED.h"
OakOLED oled;
MPU6050 mpu6050(Wire);

void setup() {
  Serial.begin(9600);
  Wire.begin();
    oled.begin();

  mpu6050.begin();
  mpu6050.calcGyroOffsets(true);
}
```

Fig 8 Arduino code



```
void loop() {
  mpu6050.update();
  Serial.print("angleX : ");
  Serial.print(mpu6050.getAngleX());
  Serial.print("\tangleY : ");
  Serial.print(mpu6050.getAngleY());
  Serial.print("\tangleZ : ");
  Serial.println(mpu6050.getAngleZ());
    oled.clearDisplay();
oled.setTextSize(1);
oled.setTextColor(1);
oled.setCursor(84, 0);
oled.println("O");
oled.setTextSize(2);
oled.setCursor(0, 0);
oled.println(mpu6050.getAngleX());
oled.setTextSize(1);
oled.setCursor(85, 10);
oled.println("AngleX");
```

Fig 9. Arduino code getting angle

To download this code, refer the link given at the bottom this page

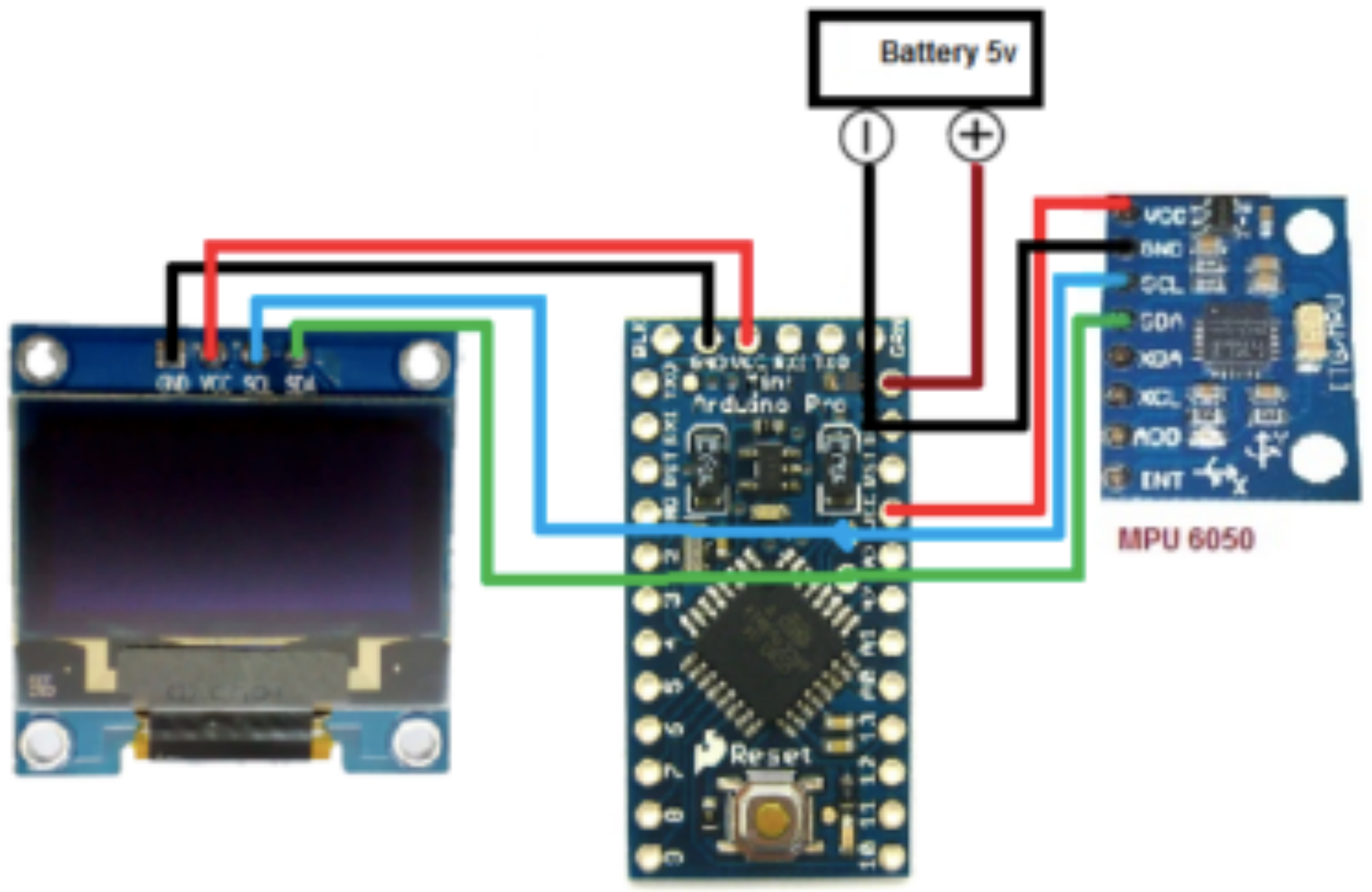


Fig 10. Connection.

Our code is done now. Let's start Connection:

Connection

Connect all the components as illustrated in Fig 10.

Arduino	Components
SCL	OLED and MPU 6050 SCL
SDA	OLED and MPU 6050 SDA
VCC	OLED and MPU 6050 VCC
GND	OLED and MPU 6050 GND

Download the Source Folder

Congrats! We have completed our project. You can move the Gyro sensor and get the angles of X,Y and Z coordinates. Put your compass on reference plane and move it to the target plane to measure the angle between two surfaces.

You can watch Video for this DIY here: https://youtu.be/pZPV4TMT7-0

20

MAKE VIDEO STREAMING CAMERA WITH RASPBERRY PI

BY ASHWINI KUMAR SINHA

Live streaming video cameras can be used for security or personal purposes. A variety of webcams, camcorders, DSLRs and mirrorless cameras for streaming live video are available in the market. You want to own one too but can't afford it! Don't worry, we will help you build one yourself at less than Rs. 2000.

Today, in this DIY project, we are going to make a live streaming camera that can be accessed over Wi-Fi. The best part of this project is that you can view the video live streaming on multiple devices like computer, tablet and phones. You can also move its camera lens wirelessly for wide-angel views. Another interesting thing is that you can set this camera in a motion detection mode to enable it to take picture or record video when motion is detected. However, you may experience video lag depending on the Wi-Fi network.

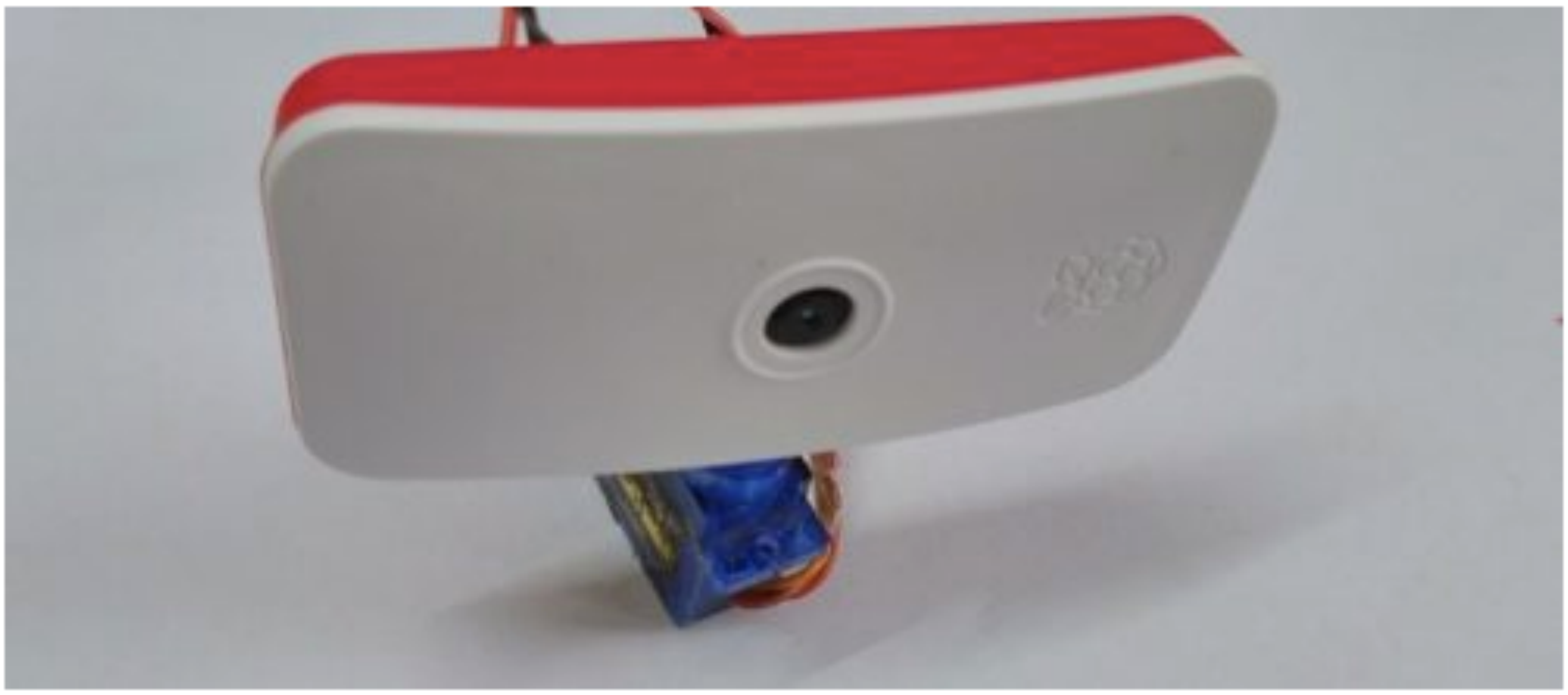


Fig 1. Prototype

We can make this project by applying two methods, which will be explained below.

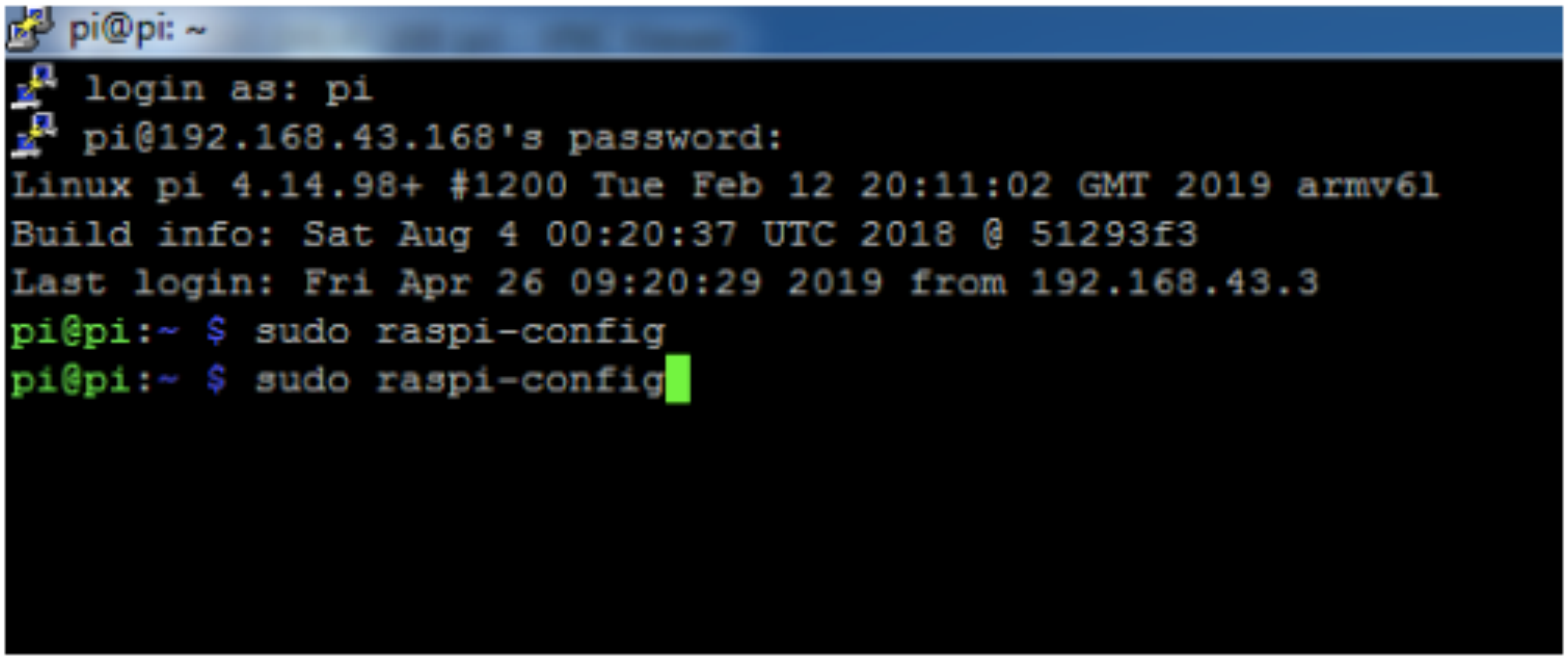
Setting Raspberry Pi for camera

First, you have to set up the Raspberry Pi for camera interface. Open the terminal and run this command “sudo raspi-config”. Then you will get a blue colour window with several options. Select the “Interfacing” option and enable the camera interface. (Refer Fig 2,3,4).

Now, your Raspberry Pi is ready for camera interface, meaning you can start streaming camera video over Wi-Fi. You can archive this using the following two methods.

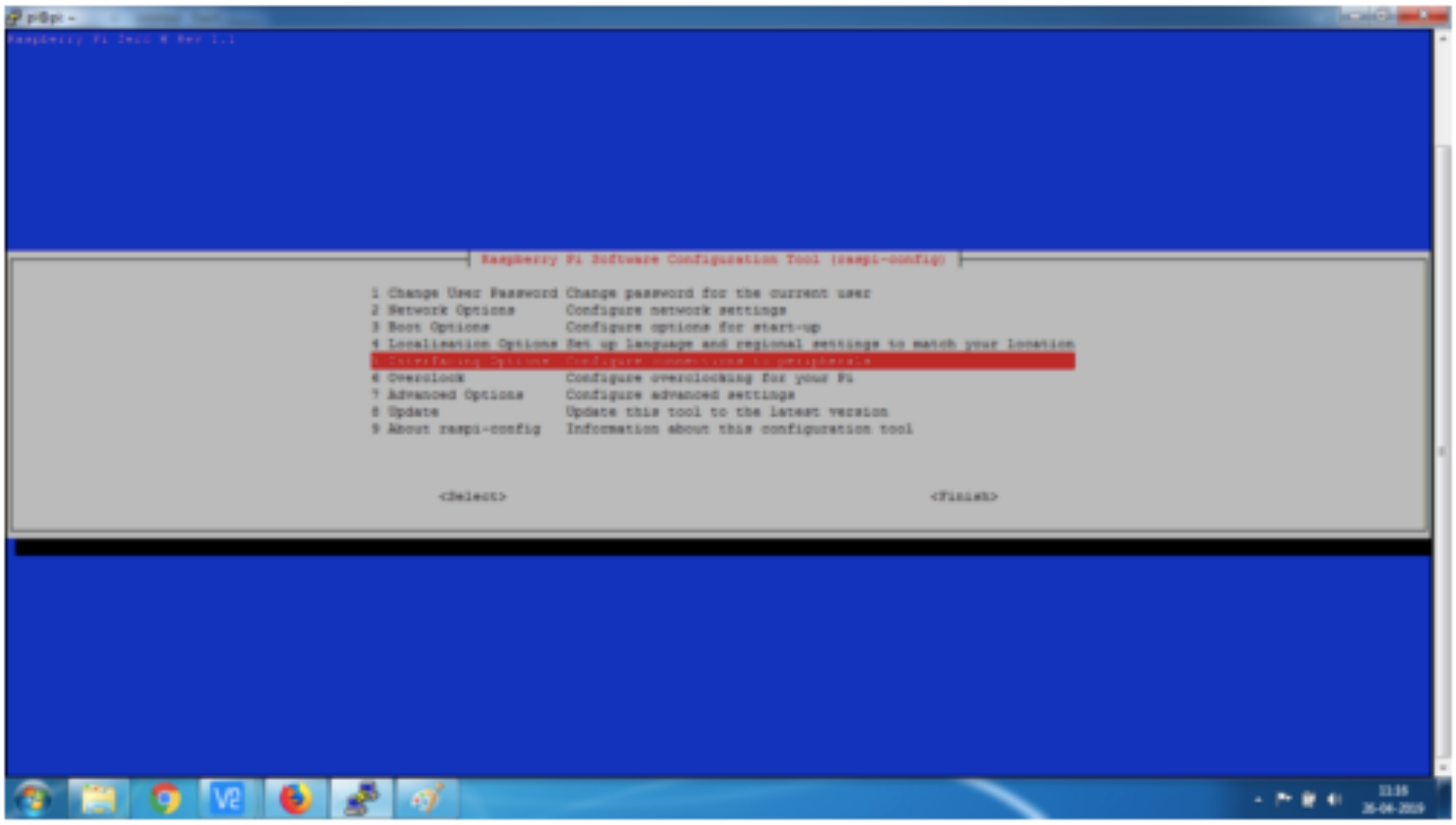
Streaming Camera Video using VLC player.

In the first method, we are going to stream the camera video using VLC player. To do this, first open the terminal window and type the following command to install VLC on your Raspberry Pi.



```
pi@pi: ~
login as: pi
pi@192.168.43.168's password:
Linux pi 4.14.98+ #1200 Tue Feb 12 20:11:02 GMT 2019 armv6l
Build info: Sat Aug 4 00:20:37 UTC 2018 @ 51293f3
Last login: Fri Apr 26 09:20:29 2019 from 192.168.43.3
pi@pi:~ $ sudo raspi-config
pi@pi:~ $ sudo raspi-config
```

Fig 2. Configuration of Raspberry Pi



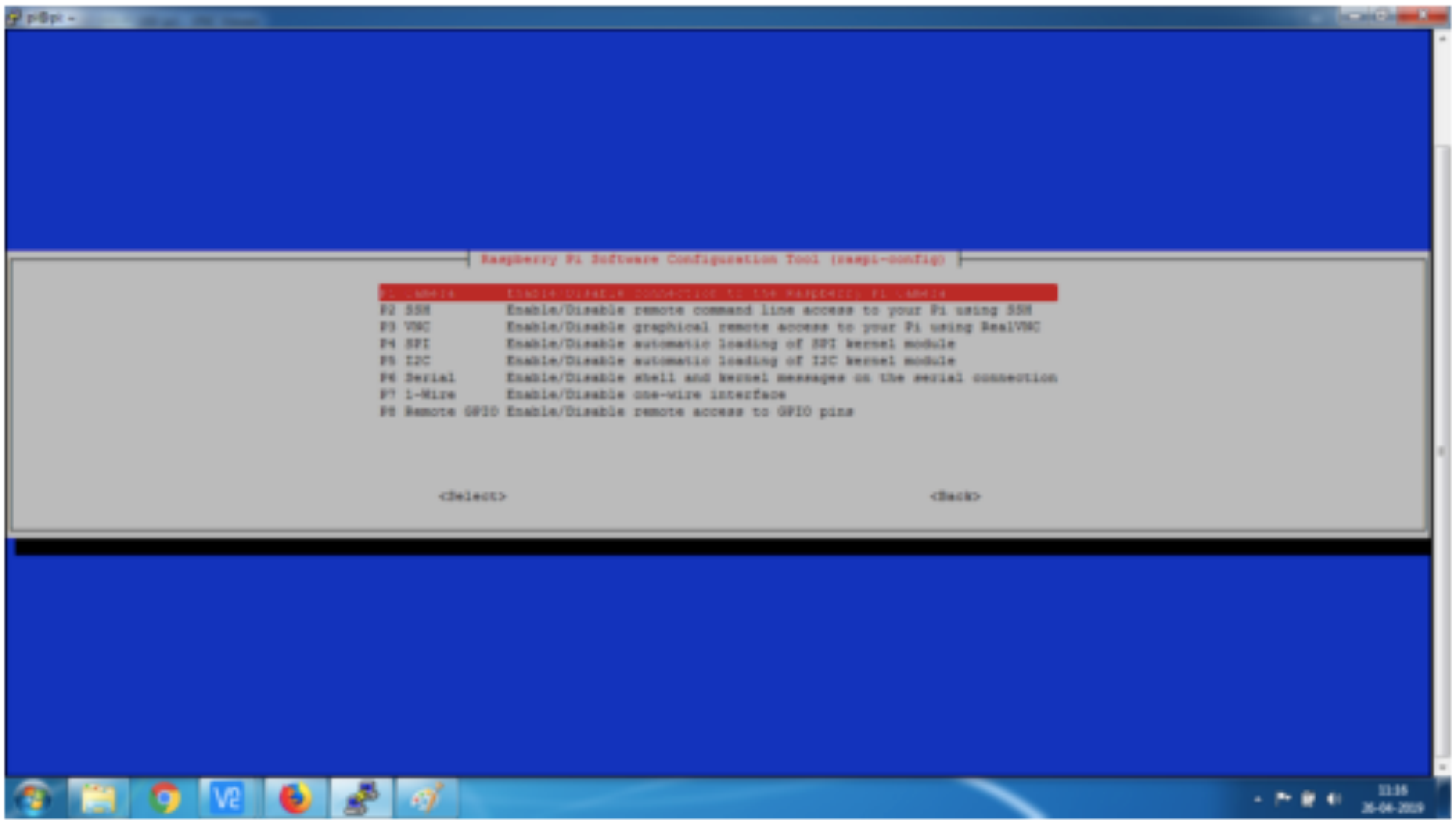
```
         Raspberry Pi Software Configuration Tool (raspi-config)

 1 Change User Password Change password for the current user
 2 Network Options      Configure network settings
 3 Boot Options         Configure options for start-up
 4 Localisation Options Set up language and regional settings to match your location
 5 Interfacing Options  Configure connections to peripherals
 6 Overclock            Configure overclocking for your Pi
 7 Advanced Options     Configure advanced settings
 8 Update               Update this tool to the latest version
 9 About raspi-config   Information about this configuration tool

        <Select>                                      <Finish>
```

Fig 3. Setting interfaces

"Sudo apt-get install vlc"

After the successful installation, you can stream your camera video using the following command

"raspivid -o - -t 0 -n | cvlc -vvv stream:///dev/stdin --sout



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

       <Select>                                    <Back>
```

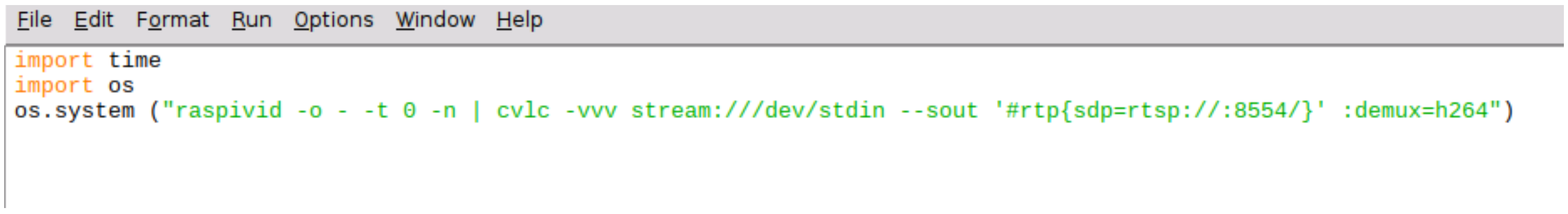
Fig 4. Setting Camera

'#rtp{sdp=rtsp://:8554/}' :demux=h264"

To obtain the output of the video streamed by the camera, connect your PC or phone to the same Wi-Fi network on which Raspberry Pi is connected. Next open the VCL player, then go on stream menu and paste the url.

rtsp://ip address of your pi:8554/

Now, open the text editor and put the following code in it



```python
import time
import os
os.system ("raspivid -o - -t 0 -n | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554/}' :demux=h264")
```

And save it as cam.py and whenever you want to live stream a video, run this program.

Streaming Video Using Motion

Open the terminal and run the following command to install motion "sudo apt-get install motion"

Then setup setting to run the camera server continuously on background

```
"sudo nano /etc/default/motion"
 start_motion_daemon=yes
```

Then make some changes in motion config as described below (Refer Fig 5).
Run this command

```
"sudo nano /etc/motion/motion.conf"
Do these changes according to your need
Stream_port=8081
Stream quality 50 # set the quality according to your need
# Allow motion to run the daemon we've set earlier
daemon on
# set the framerate of the stream (100 for higher quality)
framerate 100
# set the width and height of your video
width 640
height 480
# control de port 8080 by default
webcontrol_port 8080
# careful! don't set the stream_port just like the webcontrol port
```

When you done with the above setting, save and exit the config file using key "CTRL + X". Then hit the Y key and then press enter.

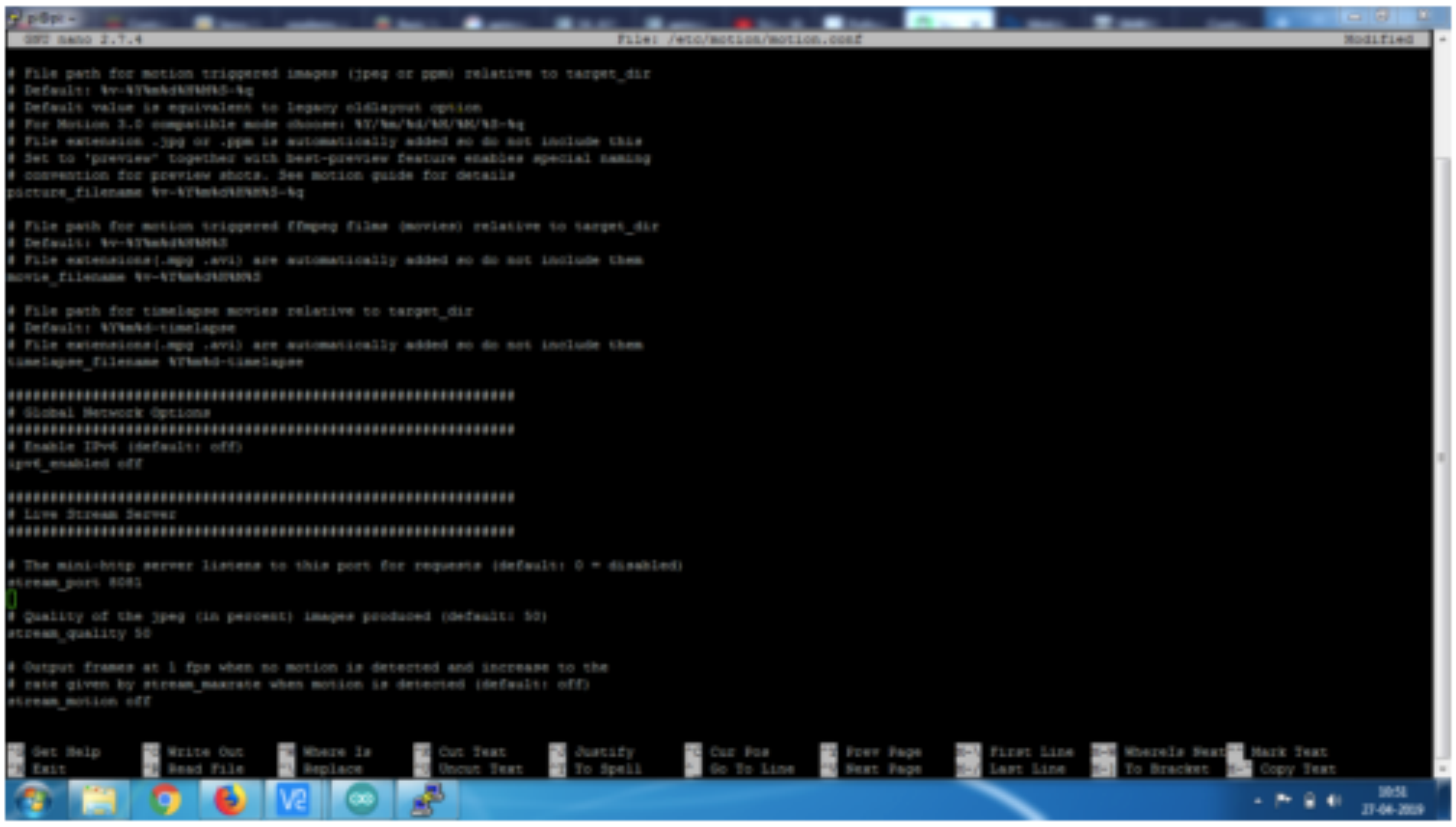
To start camera video streaming, open the terminal and run this command

```
"sudo service motion start"
"Sudo motion"
```

Now, connect your PC and Raspberry Pi on the same network. After that open any web browser and type the following url

https:// ip address of your raspberry pi:8081

Now you can get the camera video streaming on your web browser.

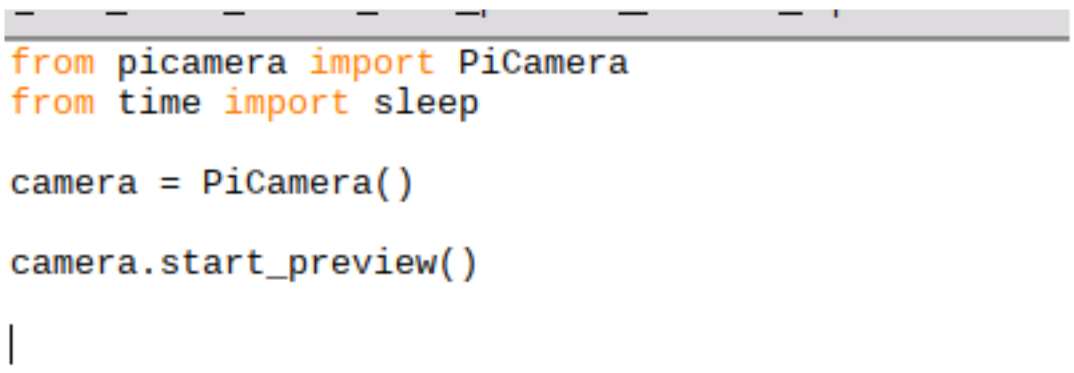


```
GNU nano 2.7.4                    File: /etc/motion/motion.conf                    Modified

# File path for motion triggered images (jpeg or ppm) relative to target_dir
# Default: %v-%Y%m%d%H%M%S-%q
# Default value is equivalent to legacy oldlayout option
# For Motion 3.0 compatible mode choose: %Y/%m/%d/%H/%M/%S-%q
# File extension .jpg or .ppm is automatically added so do not include this
# Set to 'preview' together with best-preview feature enables special naming
# convention for preview shots. See motion guide for details
picture_filename %v-%Y%m%d%H%M%S-%q

# File path for motion triggered ffmpeg films (movies) relative to target_dir
# Default: %v-%Y%m%d%H%M%S
# File extensions(.mpg .avi) are automatically added so do not include them
movie_filename %v-%Y%m%d%H%M%S

# File path for timelapse movies relative to target_dir
# Default: %Y%m%d-timelapse
# File extensions(.mpg .avi) are automatically added so do not include them
timelapse_filename %Y%m%d-timelapse

############################################################
# Global Network Options
############################################################
# Enable IPv6 (default: off)
ipv6_enabled off

############################################################
# Live Stream Server
############################################################

# The mini-http server listens to this port for requests (default: 0 = disabled)
stream_port 8081

# Quality of the jpeg (in percent) images produced (default: 50)
stream_quality 50

# Output frames at 1 fps when no motion is detected and increase to the
# rate given by stream_maxrate when motion is detected (default: off)
stream_motion off

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    ^Y Prev Page  M-\ First Line  M-W WhereIs Next  M-^ Mark Text
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line ^V Next Page  M-/ Last Line   M-] To Bracket    M-} Copy Text
```

Fig 5.Configration of motion

Cam video output either over HDMI or RCA composite

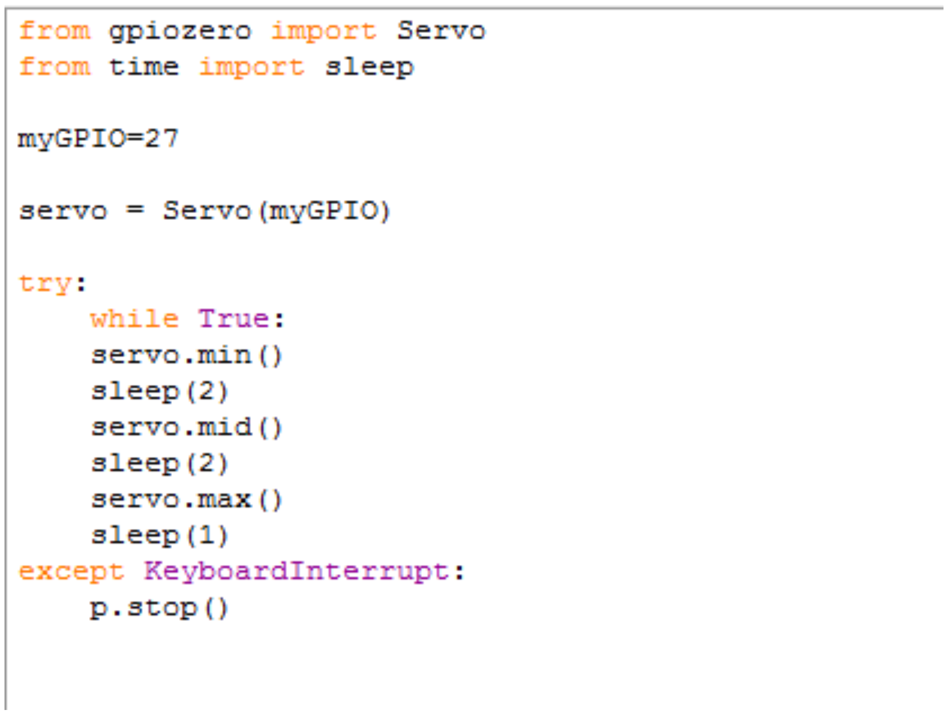
If you want to get video output like other surveillance cameras on RCA composite or on HDMI without any video lag, then create the following python code and run that code to get Raspberry Pi camera video on your TV screen.



```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
```

Giving movement to camera

Create a python program as in code snippet below and run the code for movement of servo motor.



```
from gpiozero import Servo
from time import sleep

myGPIO=27

servo = Servo(myGPIO)

try:
    while True:
    servo.min()
    sleep(2)
    servo.mid()
    sleep(2)
    servo.max()
    sleep(1)
except KeyboardInterrupt:
    p.stop()
```

As you have completed the setting and software part, it’s time to put together the whole stuff in an enclosure.

First connect the servo motor to Raspberry Pi pins as illustrated below

Motor PWM signal pin (Orange colour wire) to Raspberry Pi GPIO

Servo motor VCC to 5V power supply

GND to Power supply GND

Raspberry Pi GND to Servo Motor GND

Now, put the Raspberry Pi in camera case and fix everything according to the pics shown below -

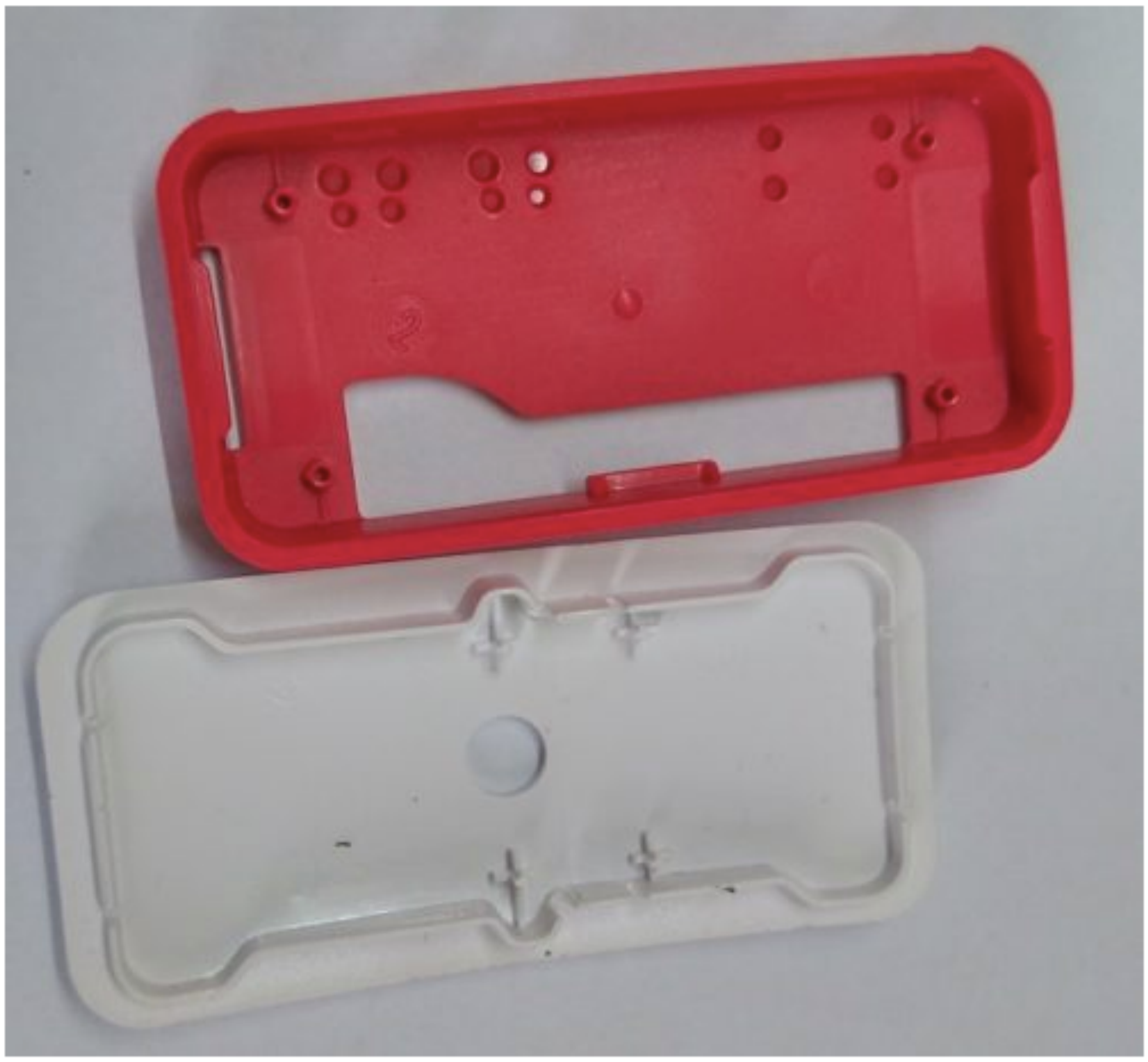


Fig 6. Raspberry pi cover.

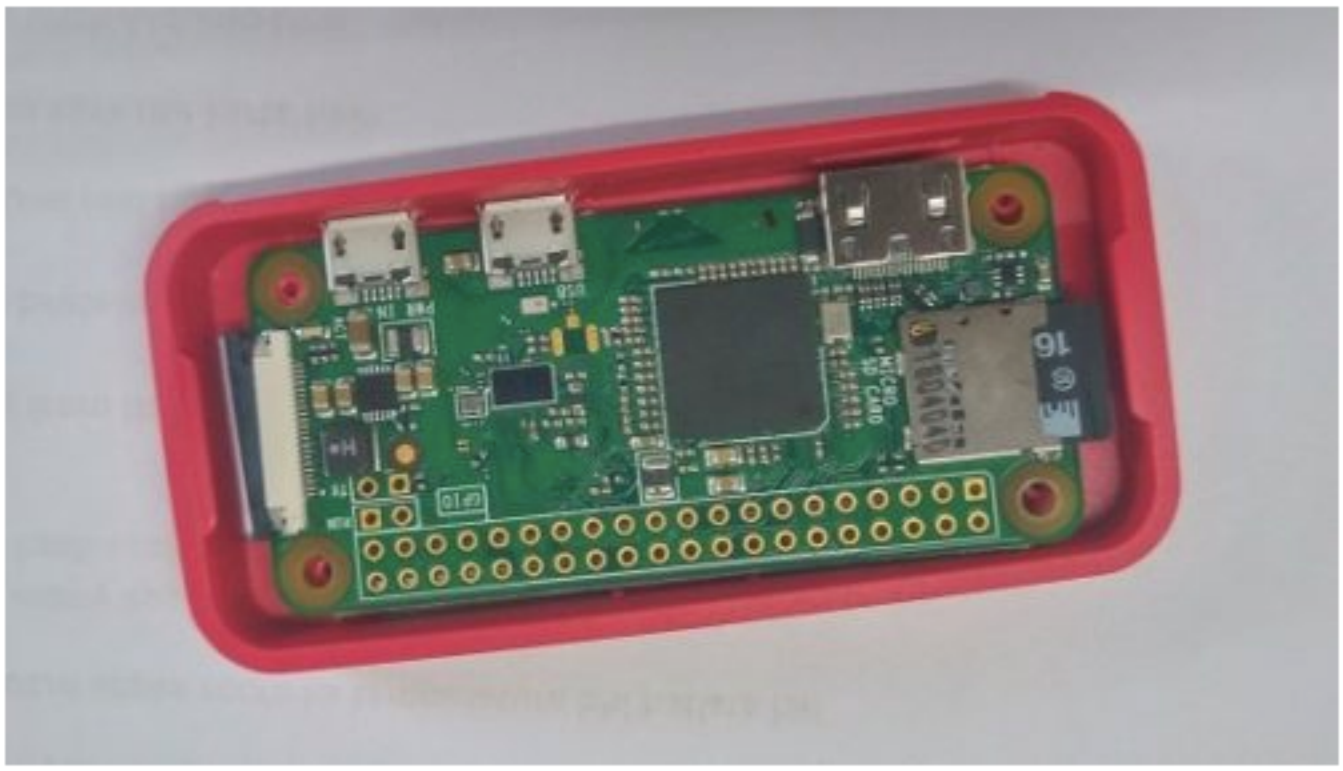


Fig 7. Raspberry pi inside case

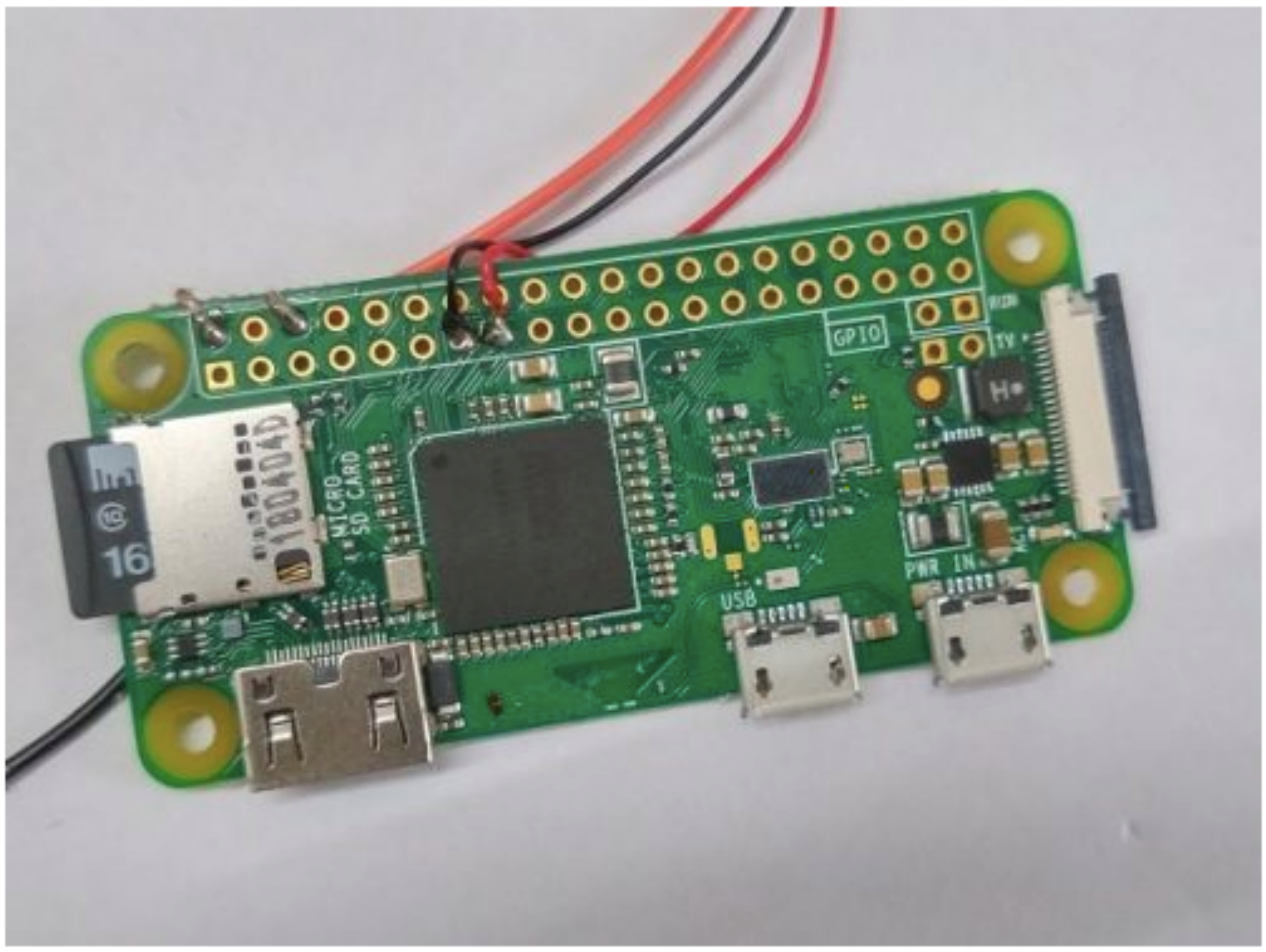


Fig 8 Solder the wires to GPIO pinouts

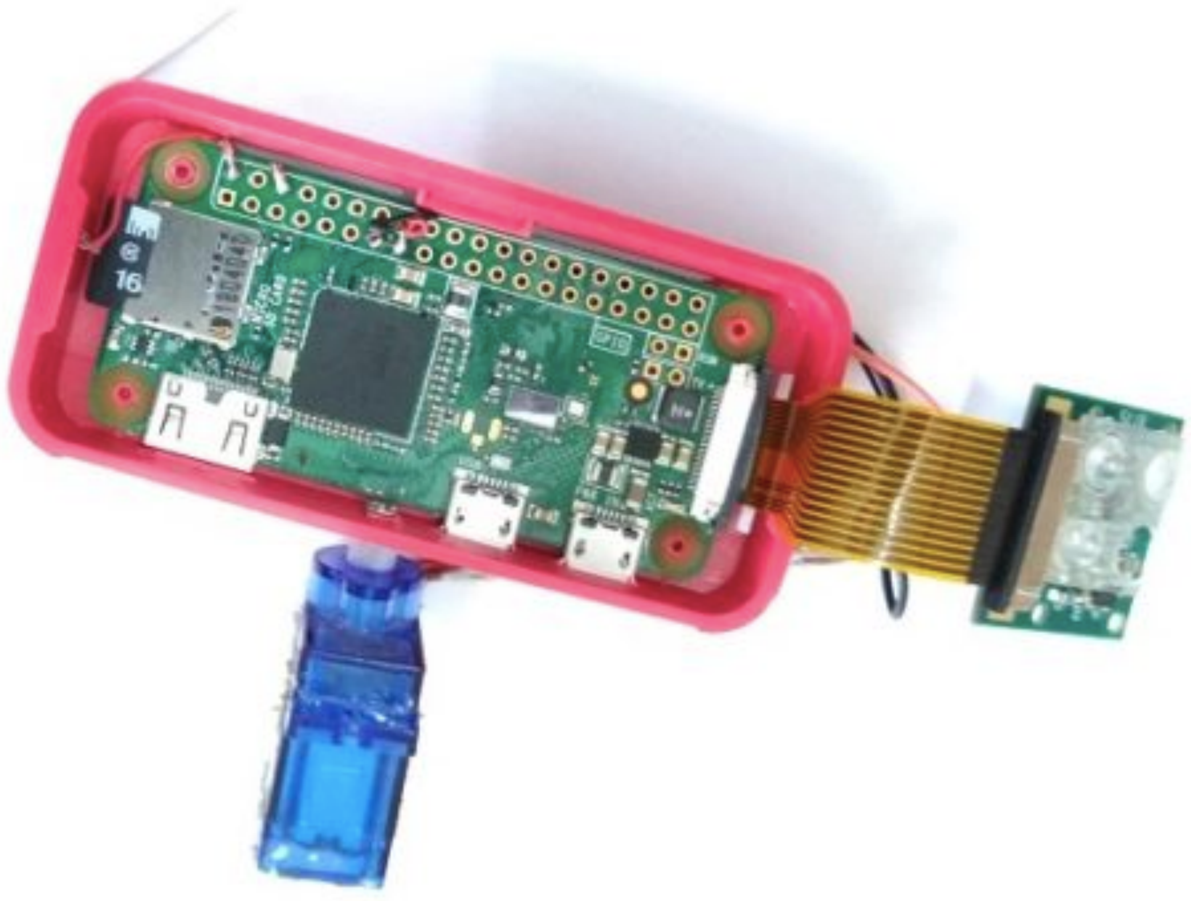


Fig 9. Mount the Raspberry pi case over servo motor

Download source code: click here

Electronicsfo
This project is very useful for physically challenged people

21

ACCELEROMETER BASED WIRELESS 3D AIR MOUSE

BY ASHWINI KUMAR SINHA

We have seen different types of Human Interface Devices (HIDs), for example mouse and keyboard. In older versions of computer mouse, optical sensors were used to detect movement relative to a surface, thus they require smooth surface to function properly. These mice can work with only two coordinates. With advancements in technology, we are now talking about 3D user interfaces ((3DUIs). So, we must start developing a 3D HID device which have the capability to work with all coordinates.

In this project we are going to make 3D wireless HID device that will give you a whole new experience of using computers and playing online games. With this device, you will be able to control computers and smartphones by just moving your hand in air.

This device can be very helpful in making 3D UI based systems. We can use this device in 3D designing as well. For instance, you can draw any design by moving your hand in the air. Further we can develop such device for 3D holographic display and in VR that gives us a 3D UI.

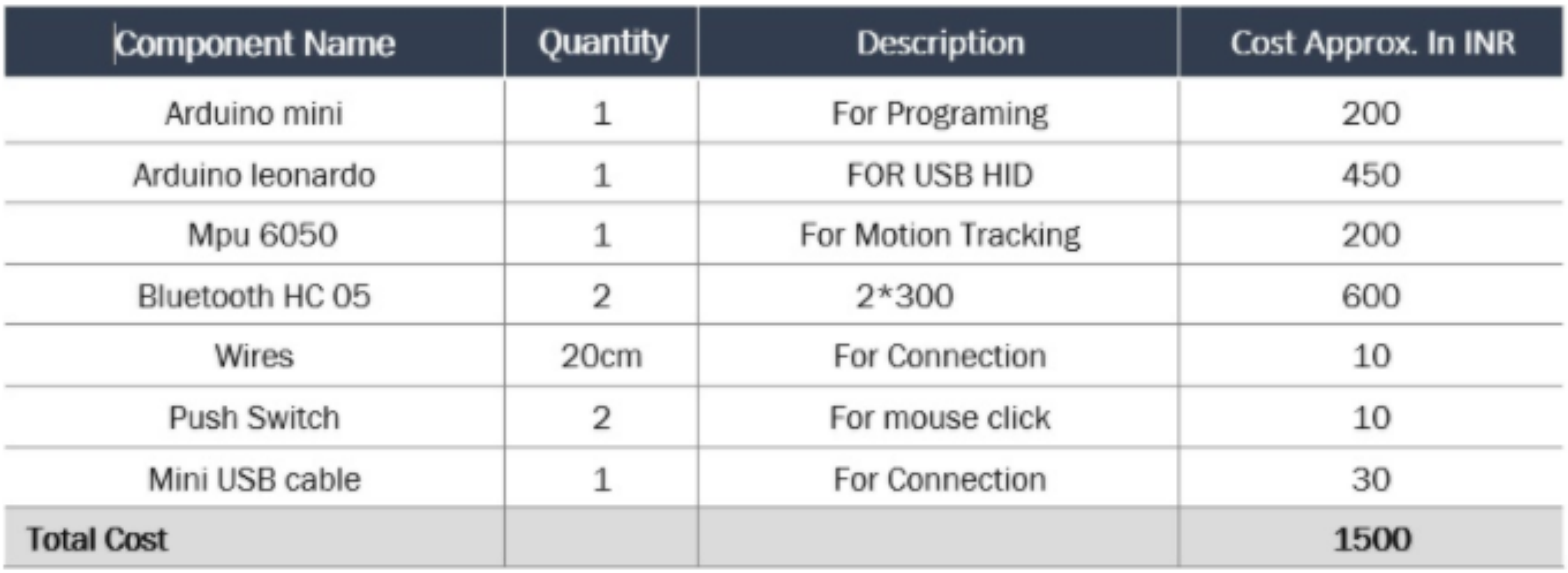
Best application of this device is that it will enable differently abled people to use computers and smartphones. People who don't have hands to operate computer or phones can now do so with the help of this device. They can wear this device on their head and operate gadgets by moving their head.

Now, let's start our Interesting project. First, we need to gather the following components -

Components:3d Mouse

- Arduino Pro Mini 5V 16-Mhz
- 2 Bluetooth HC 05
- Mpu 6050 Sensor
- Arduino Leonardo Micro (At mega 32u)
- Jumper wires
- Micro USB Cable
- 4 Push Switch

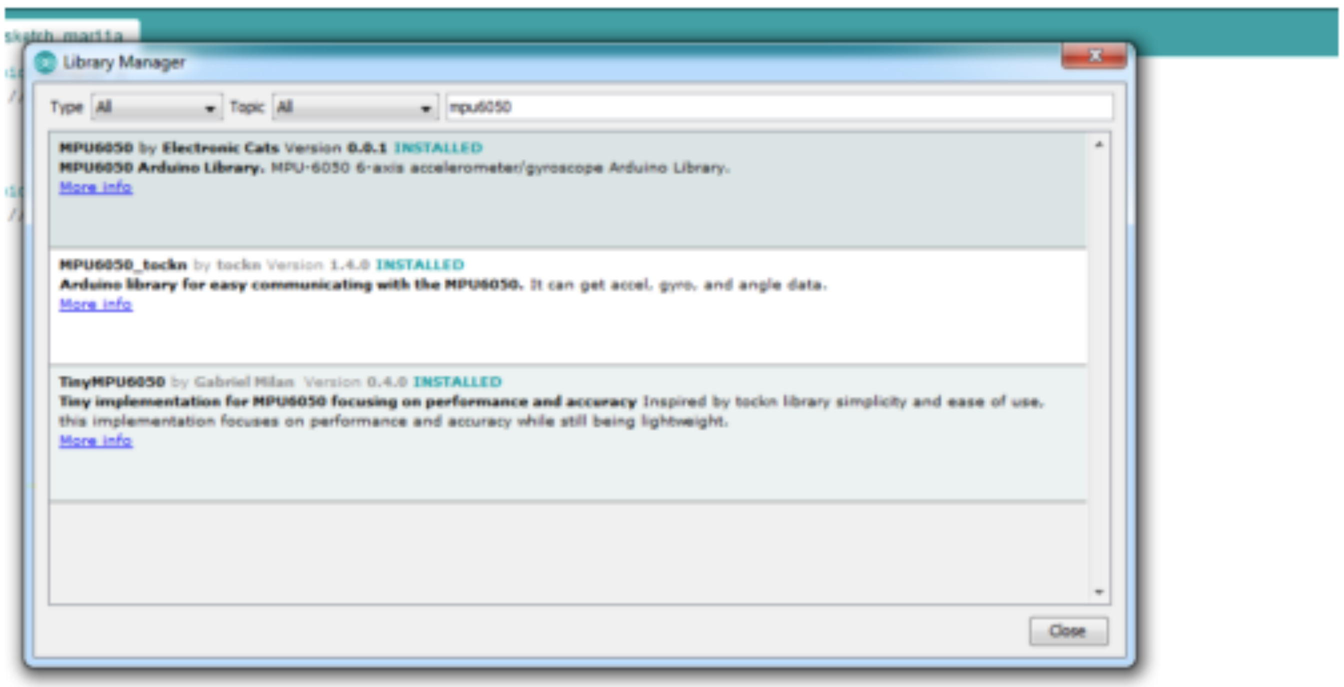
Bill of material



Component Name	Quantity	Description	Cost Approx. In INR
Arduino mini	1	For Programing	200
Arduino leonardo	1	FOR USB HID	450
Mpu 6050	1	For Motion Tracking	200
Bluetooth HC 05	2	2*300	600
Wires	20cm	For Connection	10
Push Switch	2	For mouse click	10
Mini USB cable	1	For Connection	30
Total Cost			**1500**

You also need a either FTDI Module or Arduino Uno to upload code on pro mini. Coding

First of all, we need to install an MPU 6050 library. Open library manager of Arduino IDE and search for MPU6050, then install the library - refer (Fig. 1). After this, we will set one Bluetooth as slave and another as master - follow the Bluetooth instruction on the internet to do this.



Type: All · Topic: All · mpu6050

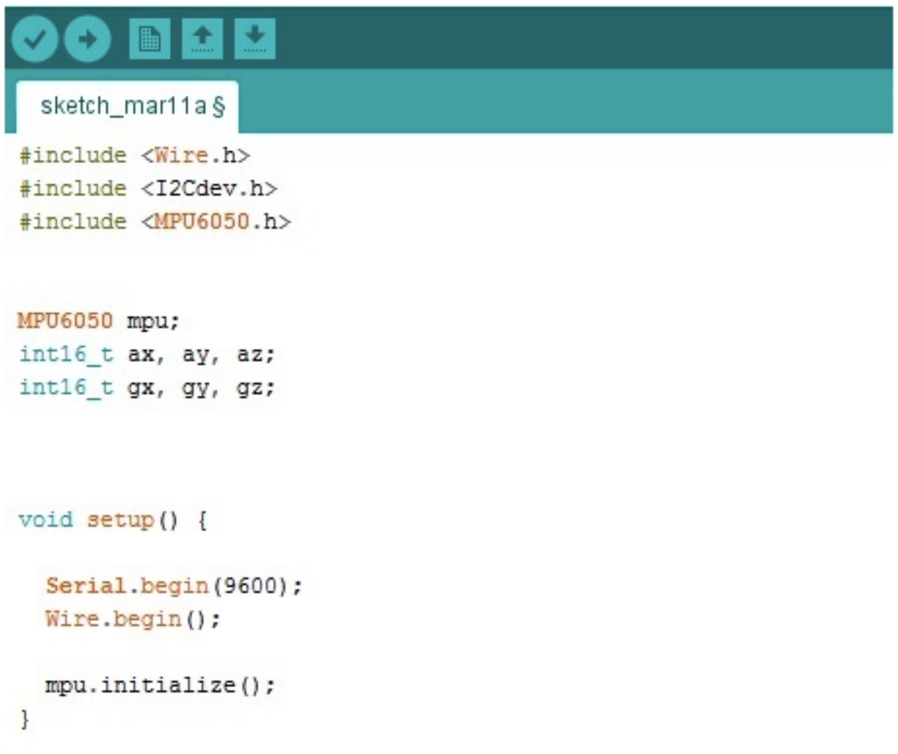
MPU6050 by Electronic Cats Version 0.0.1 INSTALLED
MPU6050 Arduino Library. MPU-6050 6-axis accelerometer/gyroscope Arduino Library.
More info

MPU6050_tockn by tockn Version 1.4.0 INSTALLED
Arduino library for easy communicating with the MPU6050. It can get accel, gyro, and angle data.
More info

TinyMPU6050 by Gabriel Milan Version 0.4.0 INSTALLED
Tiny implementation for MPU6050 focusing on performance and accuracy Inspired by tockn library simplicity and ease of use, this implementation focuses on performance and accuracy while still being lightweight.
More info

Fig. 1 Installing Library

As we have installed the library, now we can start the coding part. In the first part of coding, we will include the library for MPU6050 and set the variables for storing values of mpu6050 sensor. After that we can set up sensor and Bluetooth in set up function – for this refer (Fig. 2). I have used Baudrate 9600 in this code.



```
#include <Wire.h>
#include <I2Cdev.h>
#include <MPU6050.h>


MPU6050 mpu;
int16_t ax, ay, az;
int16_t gx, gy, gz;



void setup() {

  Serial.begin(9600);
  Wire.begin();

  mpu.initialize();
}
```

Fig .2 Arduino Mini code setting library

In the second part of coding, we will send the buttons and sensors data to another Arduino over bluetooth as in pic, refer (Fig 3).

Our Arduino mini code is ready now. So, let’s start coding for Arduino pro micro.

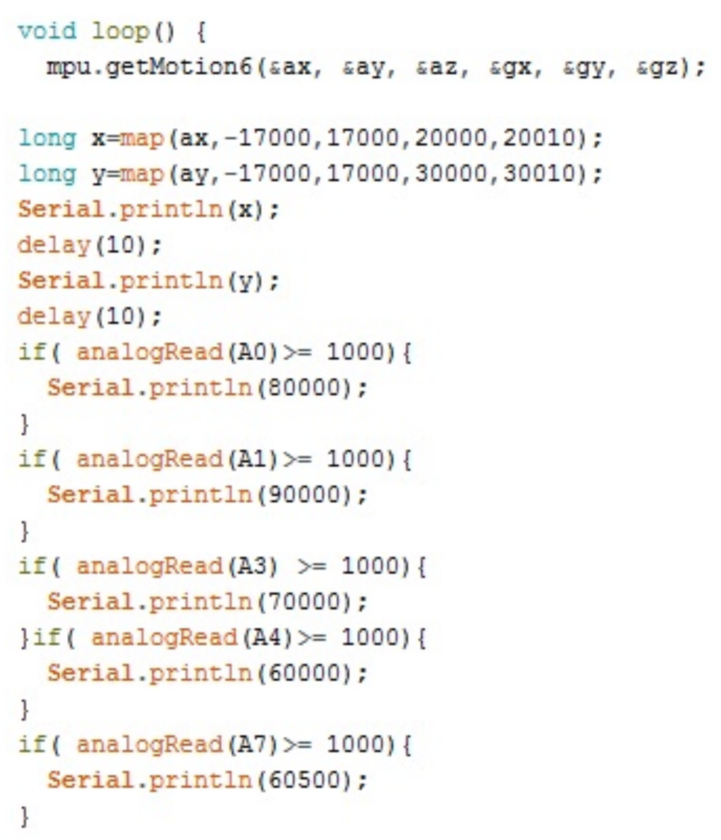
First, we will create variables to store values, then set up Bluetooth to read serial data of sensor - refer (Fig. 4).

In the third part of our coding process, we will create a loop function and get the serial data from Bluetooth. Here in this code, I have used software serial for Bluetooth connection with baudrate of 9600.

After this, check the range value of serial data and assign its value for mouse movement as in Fig. 5.

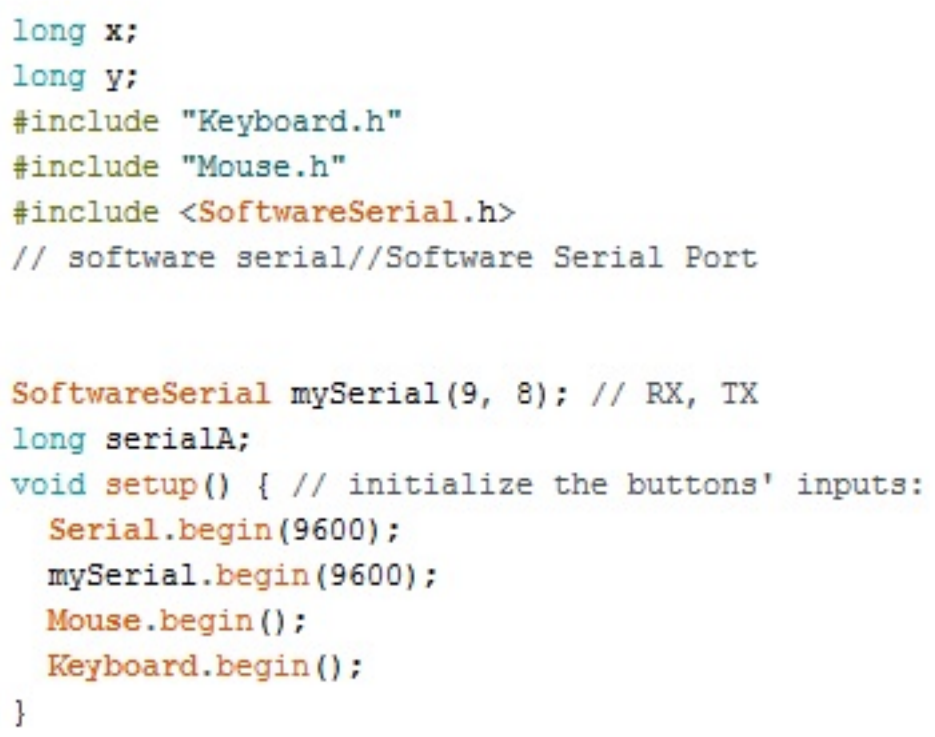
Next, we will create if () condition to check how the mouse button functions – you can refer (Fig. 6) for this. We can also add extra Keyboard shortcuts and other functions to it as per your desire. Here in this code, I have added button shortcut for copy and paste function. You can create your own custom function according to your use cases.

As we have completed the coding part, we need to connect the components.



```
void loop() {
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

long x=map(ax,-17000,17000,20000,20010);
long y=map(ay,-17000,17000,30000,30010);
Serial.println(x);
delay(10);
Serial.println(y);
delay(10);
if( analogRead(A0)>= 1000){
  Serial.println(80000);
}
if( analogRead(A1)>= 1000){
  Serial.println(90000);
}
if( analogRead(A3) >= 1000){
  Serial.println(70000);
}if( analogRead(A4)>= 1000){
  Serial.println(60000);
}
if( analogRead(A7)>= 1000){
  Serial.println(60500);
}
```

Fig. 3 Arduino Mini code getting MPU6050 data



```
long x;
long y;
#include "Keyboard.h"
#include "Mouse.h"
#include <SoftwareSerial.h>
// software serial//Software Serial Port


SoftwareSerial mySerial(9, 8); // RX, TX
long serialA;
void setup() { // initialize the buttons' inputs:
  Serial.begin(9600);
  mySerial.begin(9600);
  Mouse.begin();
  Keyboard.begin();
}
```

Fig . 4 Arduino Micro code setting library

Connections:

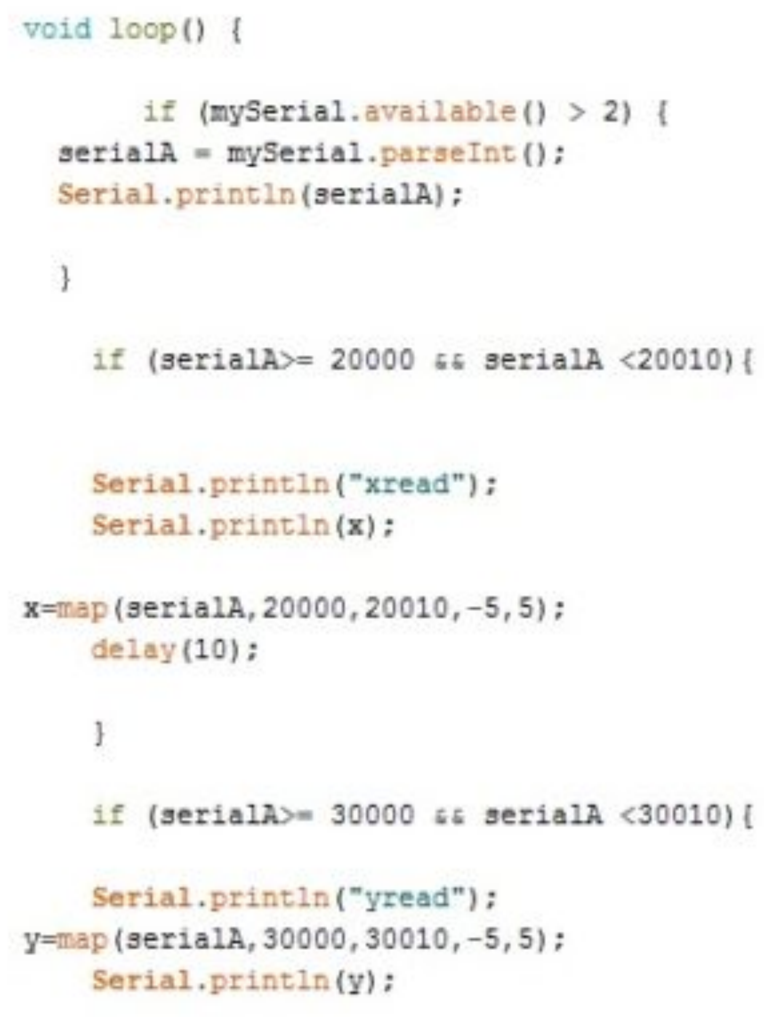
Arduino Mini Pin	Components Pin	
Arduino Mini SCL	MPU 6050 SCl	
Arduino Mini SDA	MPU 6050 SDA	
Arduino Mini VCC	MPU 6050 VCC	
Arduino Mini GND	MPU 6050 GND	
Arduino Mini TX	Bluetooth HC 05 RX	
Arduino Mini RX	Bluetooth HC 05 TX	
Arduino Mini VCC	Bluetooth HC 05 +Ve	
Arduino Mini VCC	Push Switch Pins	
Arduino Mini (A3,A0,A2,A1)	Push switch(1, 2,3,4)	

Refer Fig. 7 for connection and schematic.

Now, we will connect Arduino Pro Micro with Bluetooth - refer (Fig . 8)

Now to upload the code to Pro Mico select the board and port and and press ctrl+ U(Refer fig 9).Now upload the code to Pro mini using the board option (Refer Fig 10). Follow the instructions online to upload the code on Pro Mini. Search How to upload code on pro mini.

Congrats! The wireless HID mouse is ready for use. You can do some awesome tricks with it – you can attach it on your hand or any toy gun and enjoy the real shooting game experience. Attach the device on the tip

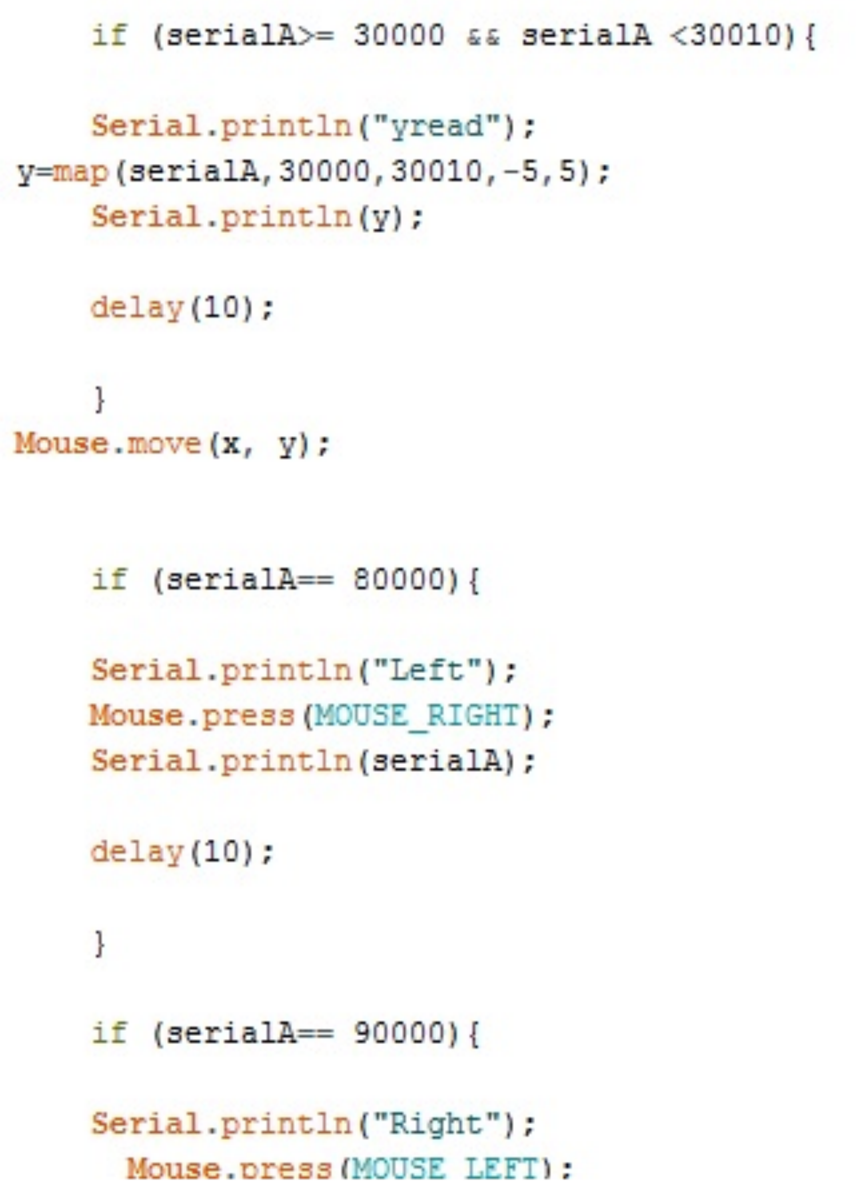


```
void loop() {

       if (mySerial.available() > 2) {
  serialA = mySerial.parseInt();
  Serial.println(serialA);

  }

    if (serialA>= 20000 && serialA <20010){


    Serial.println("xread");
    Serial.println(x);

x=map(serialA,20000,20010,-5,5);
    delay(10);

    }

    if (serialA>= 30000 && serialA <30010){

    Serial.println("yread");
y=map(serialA,30000,30010,-5,5);
    Serial.println(y);
```

Fig. 5 Checking the data to set coordinate values

of a plastic knife and you can play vegetable cutting game with it.

Note:- In the code, I have mentioned A7 pin in Arduino for a keyboard button function but not used it in connection and schematics because some Arduino mini version don't have this pin. If you have extra A7, A8 pins you can add more switch to it and set keyboard shortcut function according to your choice.



```
    if (serialA>= 30000 && serialA <30010){

    Serial.println("yread");
 y=map(serialA,30000,30010,-5,5);
    Serial.println(y);

    delay(10);

    }
 Mouse.move(x, y);


    if (serialA== 80000){

    Serial.println("Left");
    Mouse.press(MOUSE_RIGHT);
    Serial.println(serialA);

    delay(10);

    }

    if (serialA== 90000){

    Serial.println("Right");
      Mouse.press(MOUSE_LEFT);
```

Fig . 6 Setting mouse click function

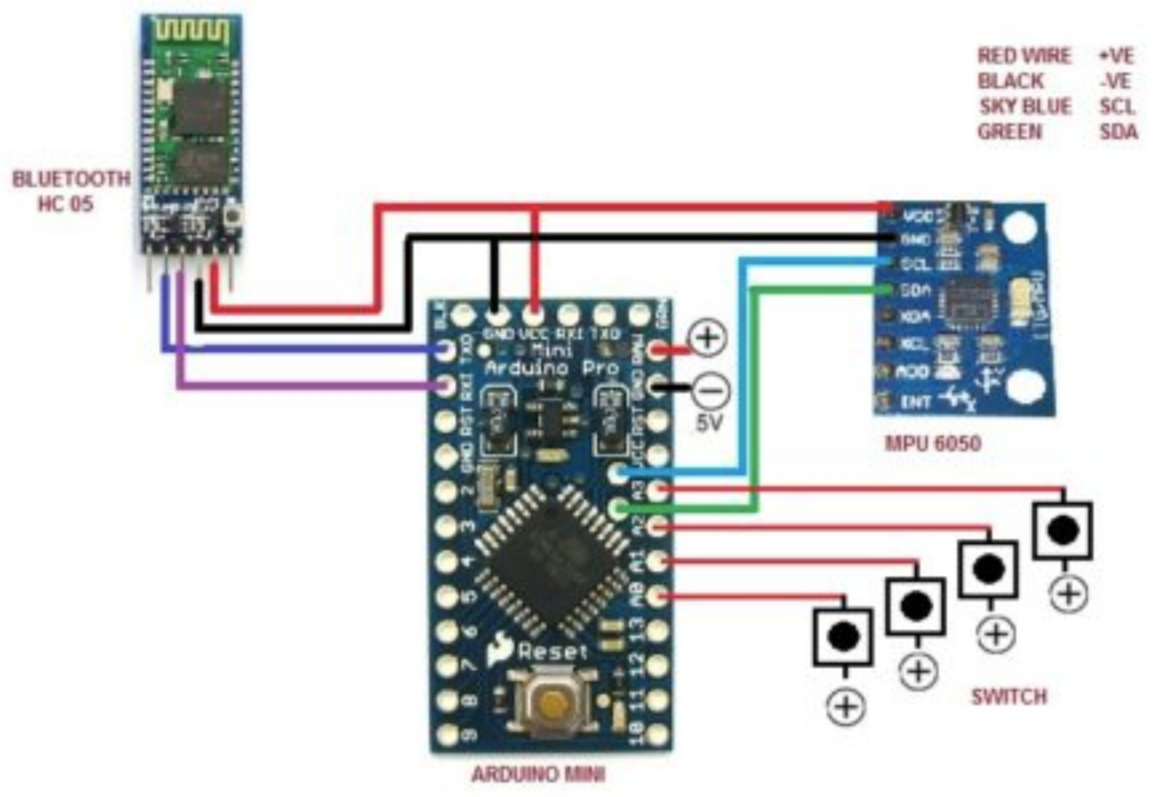


Fig .7 Connection with Arduino Pro Mini

Arduino	Bluetooth HC 05
VCC	+VE
GND	-VE
Pin 9	TX
Pin 8	RX

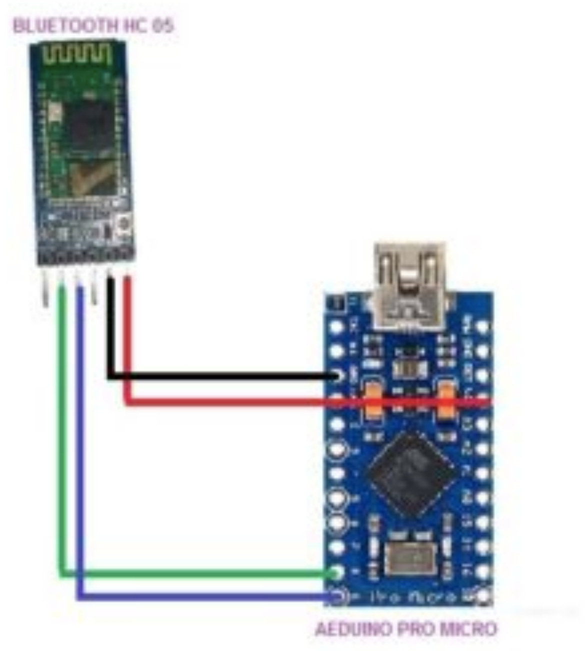
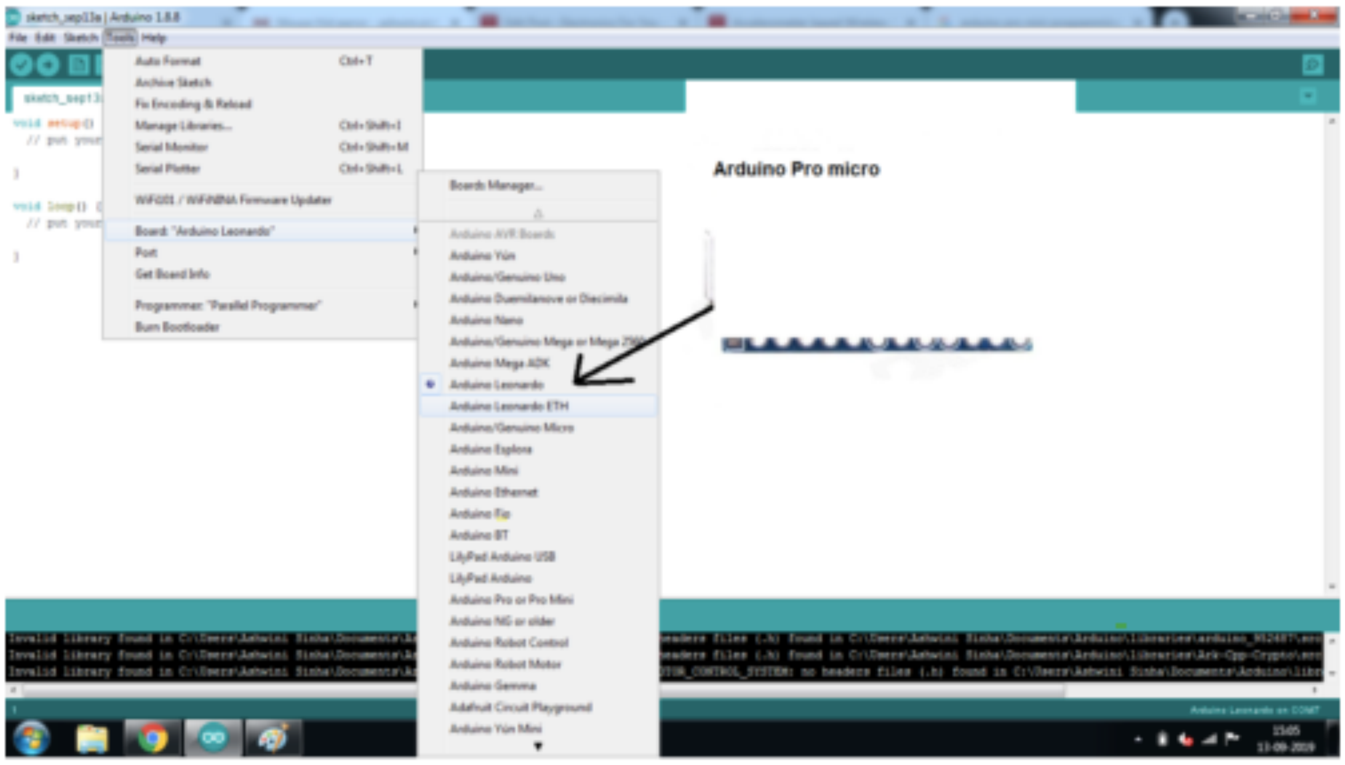


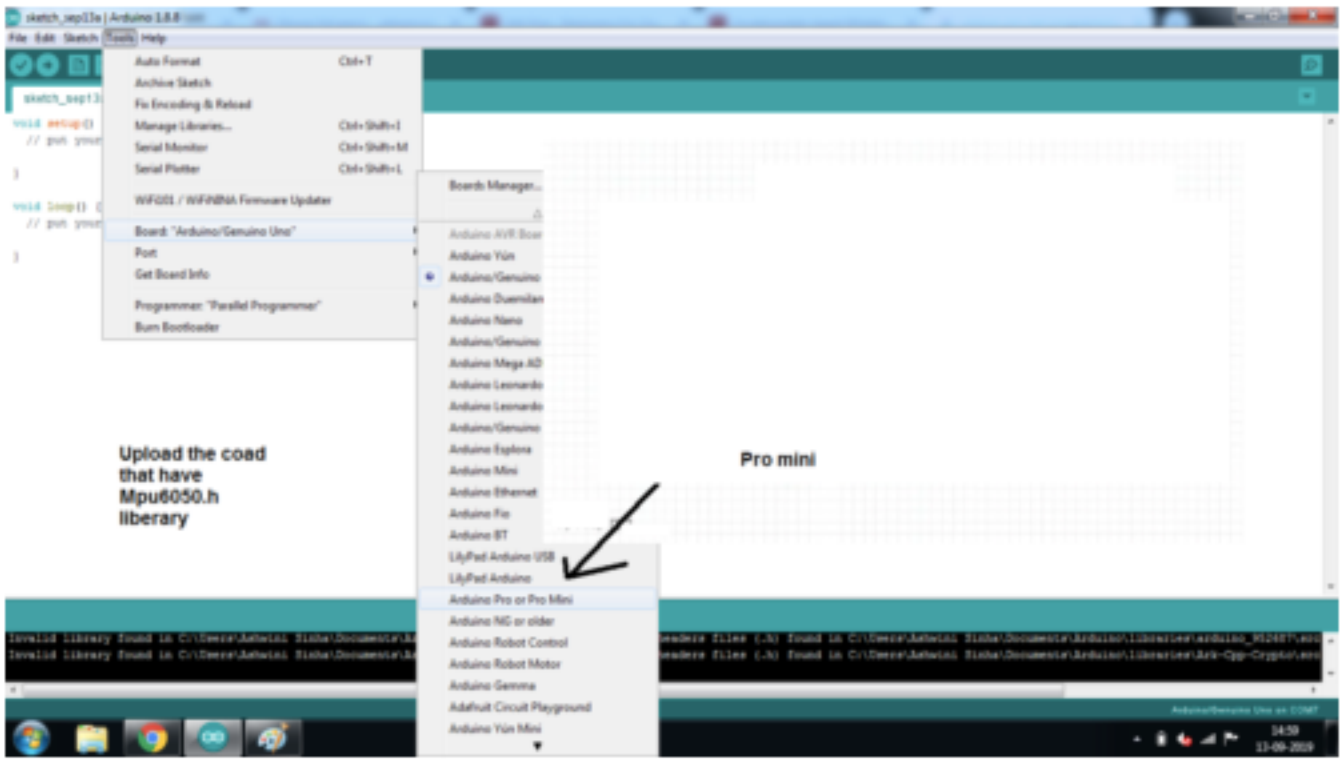
Fig .8 Arduino Pro Micro connection.



```
void setup() {
  // put your

}

void loop() {
  // put your

}
```

```
Invalid library found in C:\Users\Ashwini Sinha\Documents\A[illegible] headers files (.h) found in C:\Users\Ashwini Sinha\Documents\Arduino\libraries\arduino_952487\src
Invalid library found in C:\Users\Ashwini Sinha\Documents\A[illegible] headers files (.h) found in C:\Users\Ashwini Sinha\Documents\Arduino\libraries\Ark-Cpp-Crypto\src
Invalid library found in C:\Users\Ashwini Sinha\Documents\A[illegible]TOR_CONTROL_SYSTEM: no headers files (.h) found in C:\Users\Ashwini Sinha\Documents\Arduino\libr
```

Fig 9



```
void setup() {
  // put your
}

void loop() {
  // put your
}
```

```
Invalid library found in C:\Users\Ashwini Sinha\Documents\A...headers files (.h) found in C:\Users\Ashwini Sinha\Documents\Arduino\libraries\arduino_952487\src
Invalid library found in C:\Users\Ashwini Sinha\Documents\A...headers files (.h) found in C:\Users\Ashwini Sinha\Documents\Arduino\libraries\Arb-Cpp-Crypto\src
```

Fig 10

Download Source Folder

You can watch Video for this DIY here: https://youtu.be/ServyW75oCk

22

IOT-ENABLED AIR POLLUTION METER WITH DIGITAL DASHBOARD

BY BISWAJIT DAS

Presented here is a IoT enabled air pollution meter to monitor air quality on your smartphone using Blynk application and Arduino board. Blynk is an Internet of Things (IoT) platform to control Arduino, Raspberry Pi and the like over the Internet. In this project Blynk provides a digital dashboard on your smartphone that displays real-time air quality readings for the immediate surroundings.

Blynk is not meant for a specific board or shield. It will get you online and ready for the IoT, irrespective of whether Arduino or Raspberry Pi is linked to the Internet over Wi-Fi, Ethernet or an ESP8266 chip.

How Blynk works

Blynk can control hardware remotely. It can display sensor data, store and visualise it, among other cool things. There are three major components in the platform as given below:

Blynk app

It allows you to create amazing interfaces for your projects using various widgets.

Blynk server

It is responsible for all communication between the smartphone and hardware. You can use Blynk Cloud or run your private Blynk server locally. It is open source, and can handle thousands of devices. It can also be launched on Raspberry Pi.

Blynk libraries

Libraries are available for all popular hardware platforms. The libraries enable communication with the server, and process all incoming and outgoing commands.
When you press a button on Blynk app on the phone, the signal travels to the Blynk Cloud, where it finds its way to the hardware. It works the same way in the opposite direction, and everything happens in a blink of an eye. The concept is shown in Fig. 1.

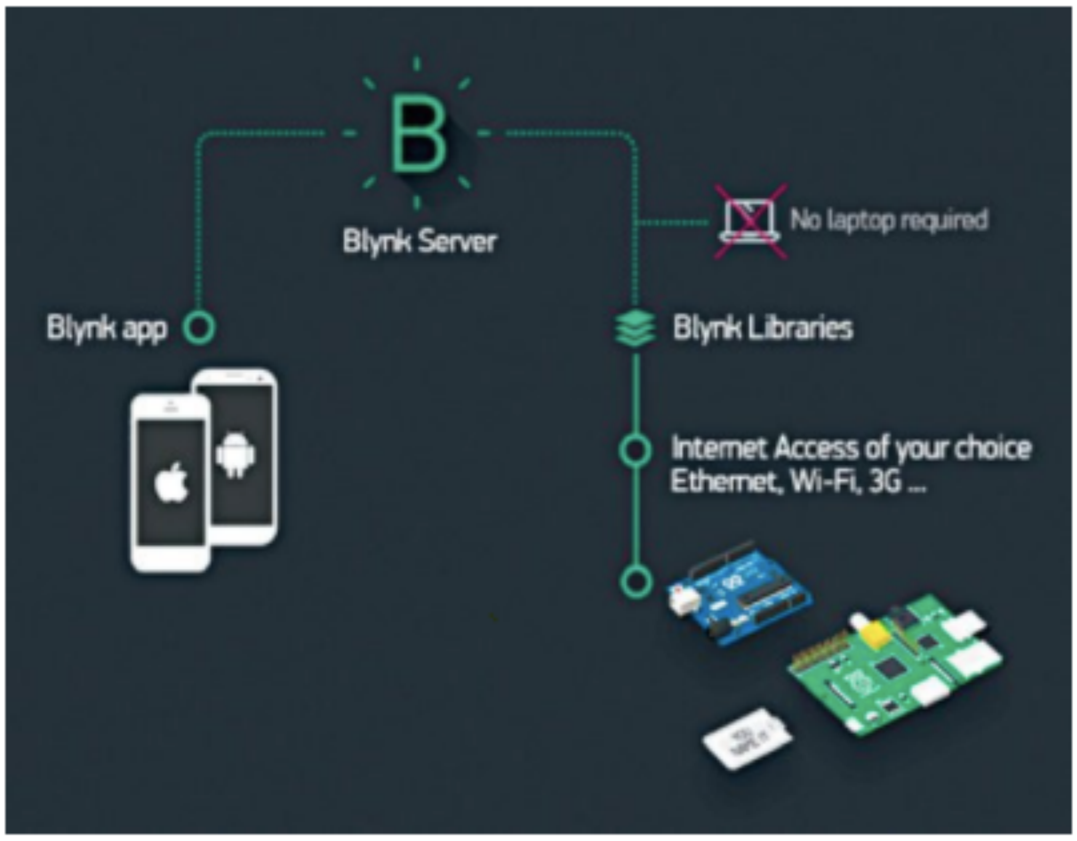


Fig. 1: Blynk gets connected with embedded hardware

IoT enabled air pollution meter

The following hardware components are used in the air pollution meter:

Controller and sensors

Arduino Uno is used as a control and measuring unit, and Arduino Uno Ethernet shield for connection to the Blynk Cloud. The sensors include nova PM sensor SDS011, gas sensor MQ135, and temperature and humidity sensor DHT11.

Smartphone and digital dashboard

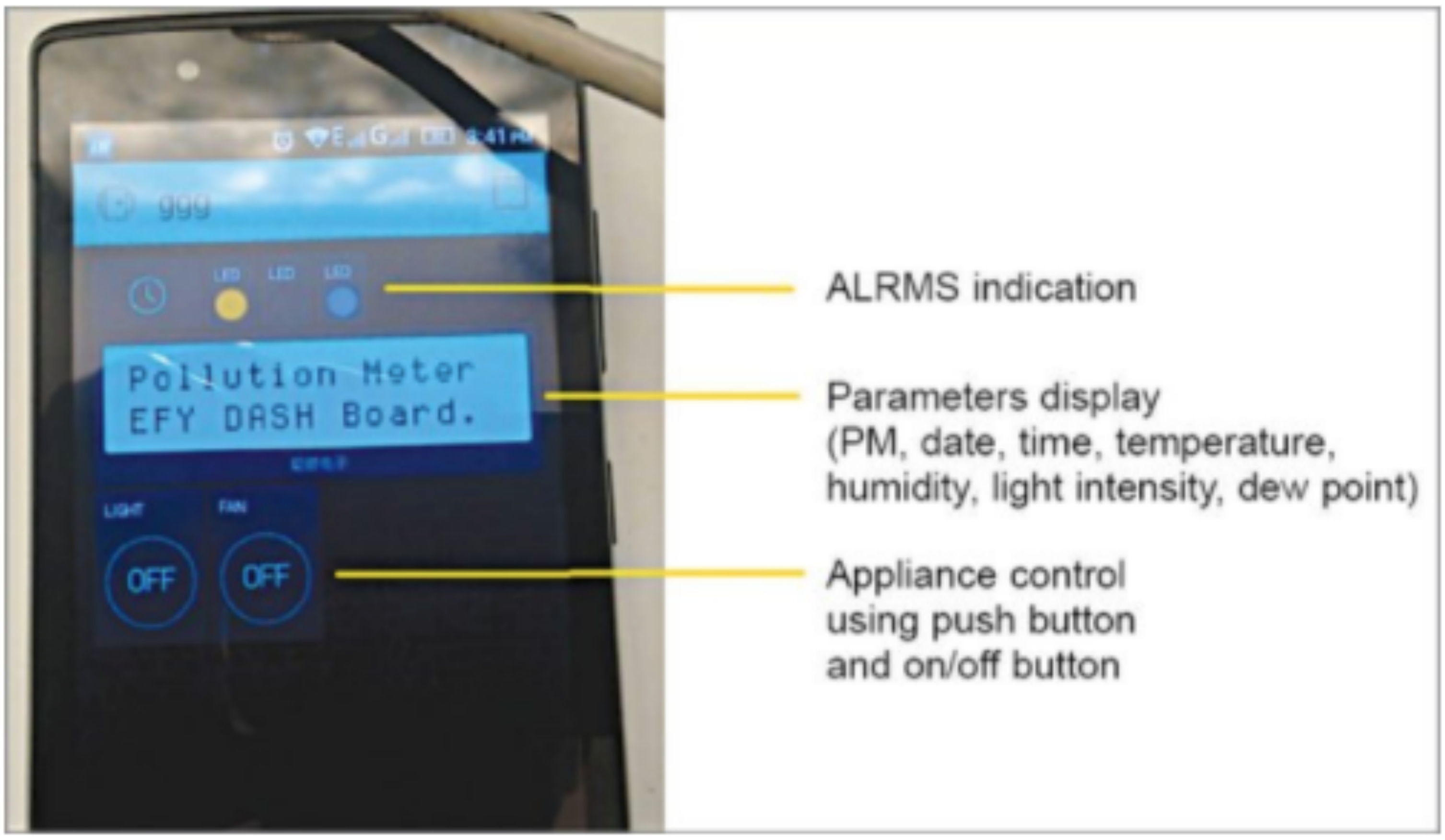

ggg
LED
LED
LED
Pollution Meter
EFY DASH Board.
LIGHT
FAN
OFF
OFF
ALRMS indication
Parameters display (PM, date, time, temperature, humidity, light intensity, dew point)
Appliance control using push button and on/off button

Fig. 2: Digital dashboard on Android phone

Android phone is used as a smartmeter. Blynk app is a well-designed interface builder. It works on iOS and Android, and provides easy-to-use widgets for the LCD, push buttons, on/off buttons, LED indication, RTC and more. Using Blynk digital dashboard features in a smartphone helps reduce the cost of the project because traditional LCD display, mechanical buttons, LEDs, RTC and so on are not required. The digital dashboard or smartmeter display is shown in Fig. 2.

Circuit and working

Circuit diagram of the IoT enabled air pollution meter with digital dashboard is shown in Fig. 3. Heart of the circuit is Arduino Uno board with Arduino shield. Other components used are voltage regulators 7805 (IC1 and IC2), temperature and humidity module DHT11 connected to connector CON3, gas sensor MQ135 connected to connector CON2, PM2.5/PM10 sensor connected to connector CON1 and a few others.

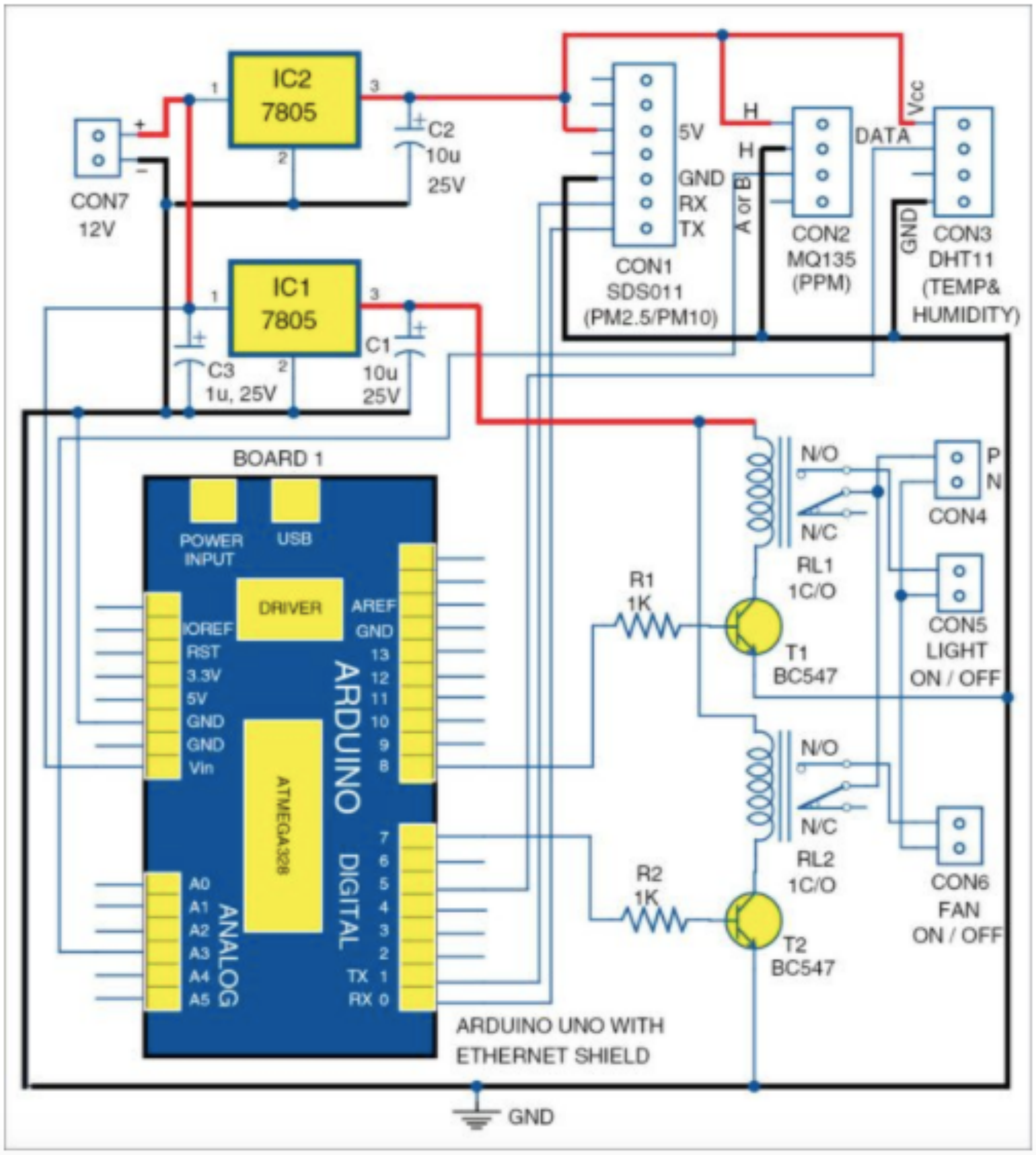


Fig. 3: Circuit diagram of the PM2.5/10 IoT enabled air pollution meter

PM2.5/PM10 sensor (SDS011)

Particle pollution, also called particulate matter (or PM), is a mixture of solid particles and liquid droplets floating in the air. Some particles are released directly from a specific source, while others are formed in complicated chemical reactions in the atmosphere.

The PM2.5/PM10 sensor connected across CON1 was developed by INOVAFIT, which is a spin-off from University of Jinan, China. It uses the principle of laser scattering in the air, and can detect suspended particulate matter concentration ranging from 0.3 to 10 microns. Data collected by the sensor is stable and reliable. SDS011 sensor is connected to UART port (TX and RX) of Arduino Uno board.

Gas sensor (MQ135)

Sensitive material of the sensor is tin-dioxide, whose conductivity increases with the concentration of gas. Change in conductivity is converted into output voltage signal, which varies corresponding to the concentration of combustible gas. MQ135 is highly sensitive to ammonia, sulphide and benzene steams, smoke and other harmful gases. It is a low-cost sensor, suitable for different applications. Output of the gas sensor is connected to analogue input pin A3 of Arduino Uno board through connector CON2.

Temperature and humidity sensor (DHT11)

This composite sensor contains calibrated digital signal outputs of temperature and humidity. Connected to connector CON3, it includes a resistive-type humidity measurement component and an NTC temperature-measurement device. Its output pin is connected to digital pin 5 of Arduino Uno board. It is a relatively inexpensive sensor for its performance.

Additional features

Additional alarm indication and control through relays RL1 and RL2 have been provided. The two appliances (light and fan) can be connected to connectors CON5 and CON6, but you can add more as per your requirement. Connector CON4 is used for connecting 230V AC mains to drive the lamp and fan connected at CON5 and CON6, respectively.

Power supply

A 12V battery or supply is connected to connector CON7, which is regulated to 5V using 7805 regulators (IC1 and IC2). When the system is powered on, the device takes one minute to preheat the gas sensor before it is ready for operation. The whole sensor circuit can be powered by 9V or 12V adaptor or battery. Here, IC1 and IC2 have been used for driving various parts of the circuit.

Enclose the whole sensor circuit (Fig. 3) in a suitable box, which should be placed where you want to monitor air quality or pollution.

Ethernet shield

Arduino Ethernet shield allows you to easily connect Arduino to the Internet. This shield enables Arduino to send and receive data from anywhere in the world, with an Internet connection.

Digital dashboard setup

Step 1. Mount Ethernet shield on Arduino Uno and connect Board1 to PC using a USB cable.

Step 2. Change the IP address in IPAddress IP (your IP address) in Arduino sketch ethernetclient.ino. This sketch/program code can be downloaded from the end of the article.

Compile and upload the code into Arduino Uno from Arduino IDE. This code shows you how to make an HTTP request using an Ethernet shield. It returns a Google search for the word Arduino. Results of this search are viewable as HTML through Arduino’s serial window.

Step 3. Compile ethernetserver.ino sketch and upload it to Arduino Uno board. Change your IP address in IPAddress IP (your IP address) in the sketch. In this example, use your Ethernet shield and Arduino board to create a simple Web server. Using the Ethernet library, your device will be able to answer an HTTP request with the Ethernet shield.

After navigating to the Ethernet shield's IP address, Arduino will respond through HTML browser and will be ready to accept input values from analogue pins (A0 through A5) of Board1.

Step 4. Connect your mobile with Wi-Fi. Download and install Blynk app from Google Play store. Then, create a new Blynk account (Fig. 4). This account is separate from the accounts used for Blynk Forums, in case you already have one.

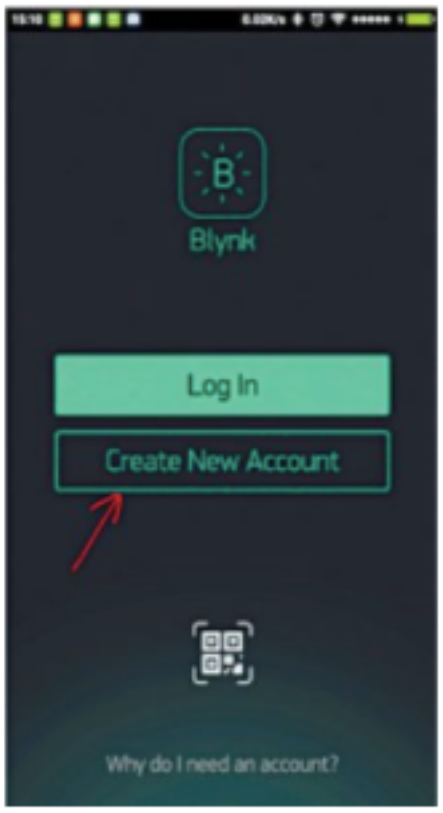


Fig. 4: Creating a new account in Blynk

An account is needed to save your projects and have access to these from multiple devices anywhere in the world. It is also a security measure. We recommend using a real email address because it will simplify things later.

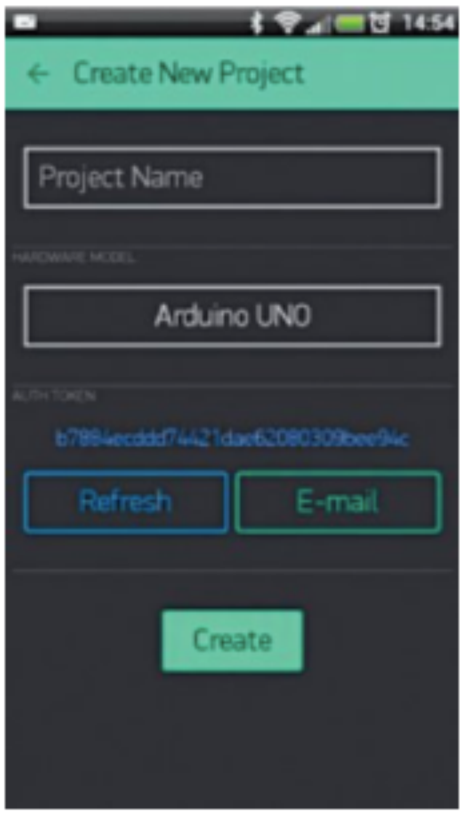


Fig. 5: Creating a new project window

Step 5. After you have successfully logged into your account, start by creating a new project (Fig. 5), and give it a name.

Step 6. Select the hardware model you intend to use. In this case it is Arduino Uno (Fig. 6).

Step 7. Authorisation (or Auth) token is a unique identifier that is needed to connect the hardware to your smartphone. Every new project you create will have its own Auth token (Fig. 7). Click email button and the token will be sent to the email address you used for registration. Use this token in auth[] = "your token" in pollution.ino file.

Download Blynk library (Blynk_v0.3.4.zip) from https://github.com/blynkkk/blynk-library/releases/tag/v0.3.4, and include it in

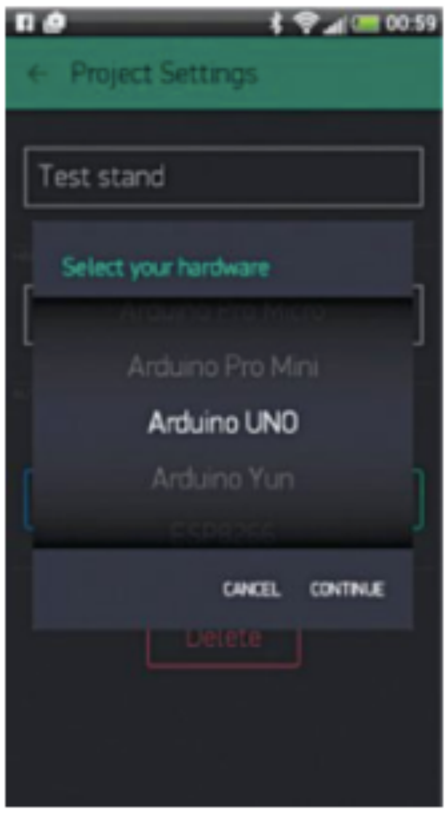


Fig. 6: Selecting the hardware

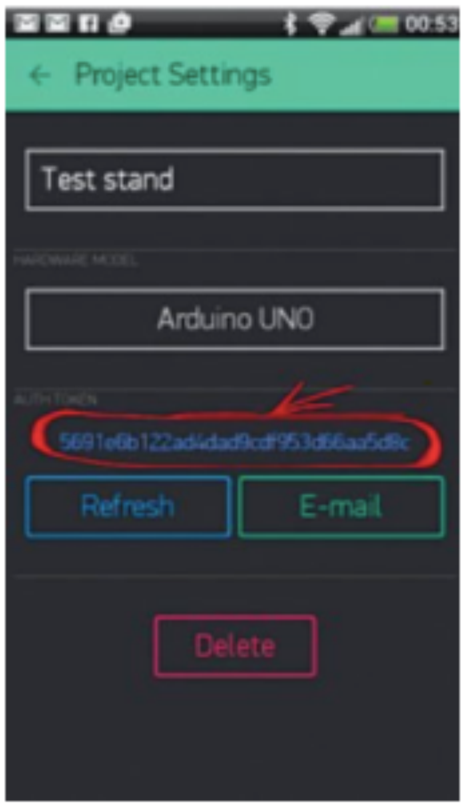


Fig. 7: Typical authorisation token

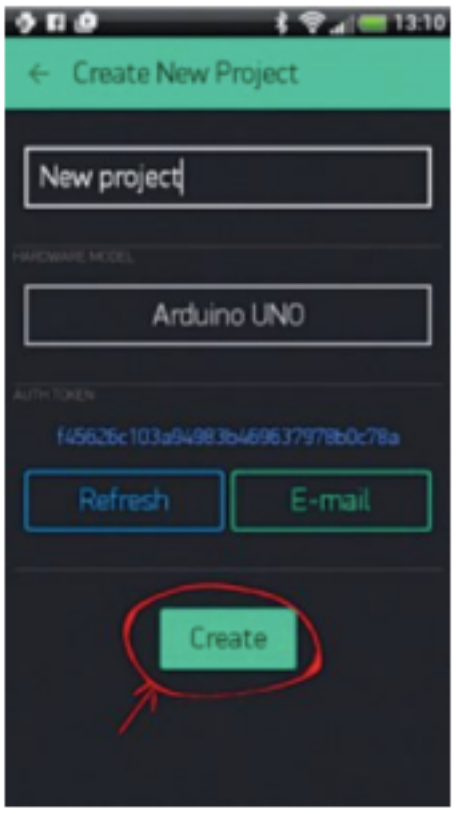


Fig. 8: Pressing Create button

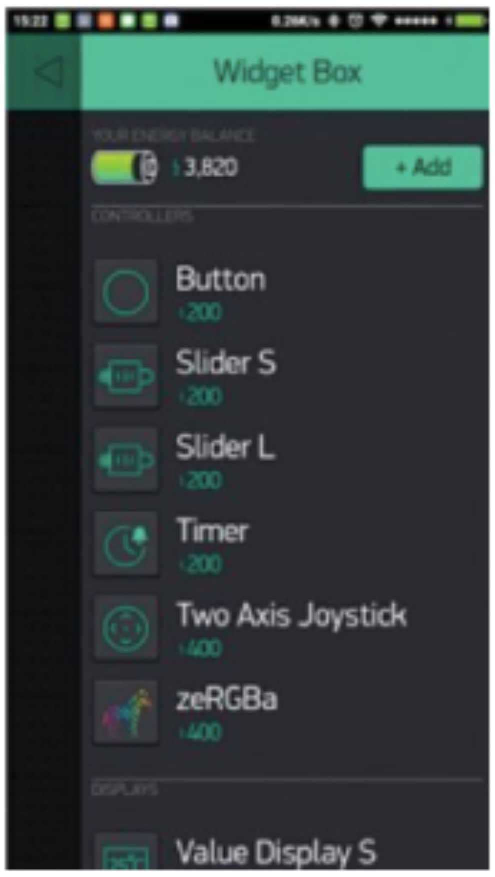


Fig. 9: Widget Box window

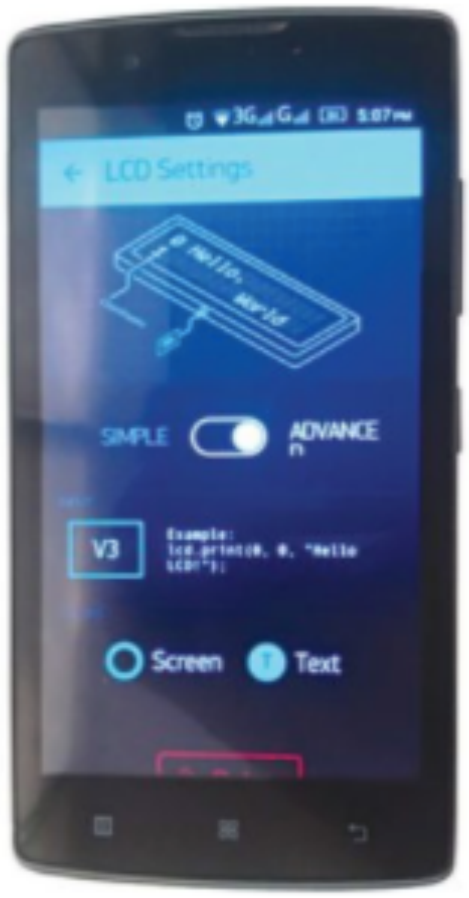


Fig. 10: LCD widget setting

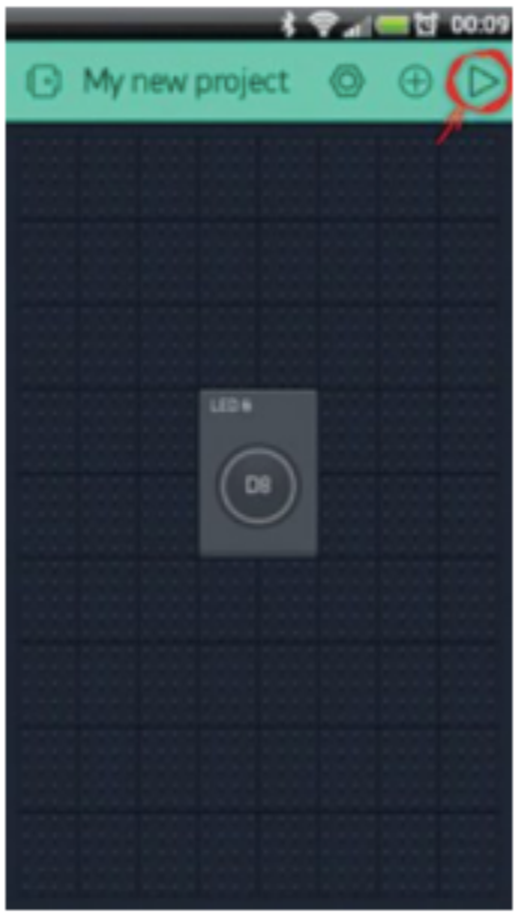


Fig. 11: Play button

Arduino IDE. Compile and upload pollution.ino code to Arduino board. Step 8.

Press Create (Fig. 8).

Step 9. Your project canvas is empty at the moment. Tap anywhere on the canvas to open the widget box; all available widgets are located here. Add widgets (Fig. 9).

Add the following widgets in the settings and make the screen look like a digital dashboard as shown in Fig. 2. This includes adding the LCD (Fig. 10), LED, on/off switch, pushbuttons and RTC widgets.

Step 10. Run the project. When you are done with the settings, press Play (Fig. 11). This will switch the display from edit to play mode where you can interact with the hardware. While in play mode, you will not be able to drag or set up new widgets. Press Stop and get back to edit mode. In play mode, you will find the screen shown in Fig. 12.

EFY notes: Use different Wi-Fi networks for your Android mobile and Arduino sensor board for a proper IoT based test.

Download source folder

23

IOT-BASED SMART AGRICULTURE SYSTEM

BY ASHWINI KUMAR SINHA

Most of us have a small garden, farmland or a plantation area. Our busy schedule, however, doesn't allow us to maintain it well. But with the use of technology, we can easily achieve it.

So, today we are going to make an IoT-based smart farming system that can monitor soil moisture. Through this device, we will be able to automatically irrigate a piece of land and wirelessly spray fertilizers and pesticides using our phone while we are busy with other work.

So, let's first gather the components for the project.

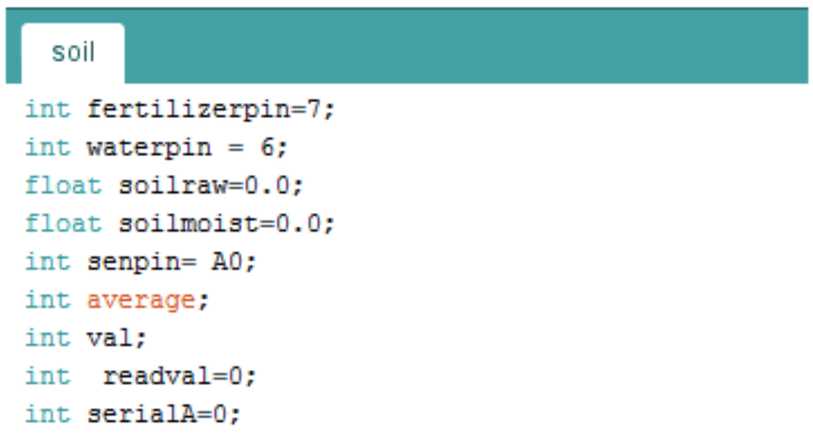
Bill Of Materials

Component Name	Quantity	Description	Cost Approx. In INR
Arduino mini/nano/micro	1	Programming	200
Soil Moisture Sensor	1	Sensors	90
Bluetooth Hc 05	1	For Communication with app	300
1 Relay Module	1	2 Channel DPDT Relay	150
Solenoid Valve	2	For Irrigation	120
Pipe and hose	-	For Irrigation/ Fertilizers	300
Wires	-	Connection	20
Total Cost			**1180**

Coding

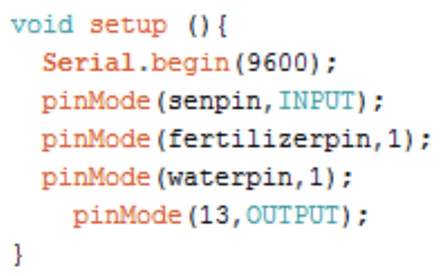
After that, we create a loop function where we create several 'if' conditions. Then, we have to create a 'for' function that will read the soil sensor data upto 160 times. Out of that, an average reading will be obtained which we can map (i.e. convert) in the range of 100 to get the moisture level output of soil in the range from 0 to 100 per cent.

The earlier created 'if' condition will check the moisture percentages. In case that percentage is very low, then irrigation will



```
int fertilizerpin=7;
int waterpin = 6;
float soilraw=0.0;
float soilmoist=0.0;
int senpin= A0;
int average;
int val;
int  readval=0;
int serialA=0;
```

Fig1.



```
void setup (){
  Serial.begin(9600);
  pinMode(senpin,INPUT);
  pinMode(fertilizerpin,1);
  pinMode(waterpin,1);
    pinMode(13,OUTPUT);
}
```

Fig 2.

automatically take place as per needed. (Refer fig 3,4,5)

Now, let's create the app

App Building

First, we create a layout of the app. This app uses an extension plugin that visualises the soil data in the form of a circular ring.

You can download the .aia file attached here that includes all the pre-made features for that. Now, import it to your Kodular creator. In this app, we have added the following components using which, we have created an app layout (Refer fig 6,7)

Components:-

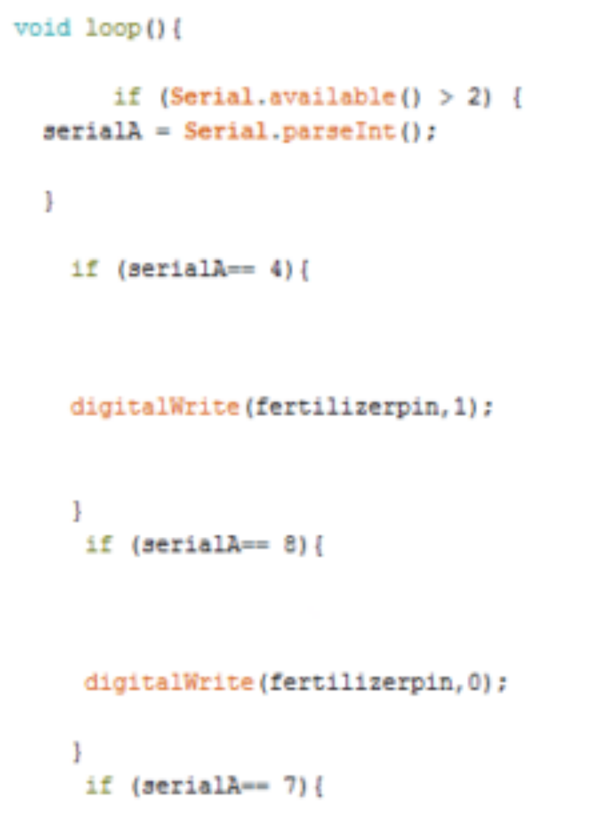
- Circular bar extension
- Bluetooth Client
- List picker
- 4 Buttons
- 2 Text View

Now, join the app code blocks and define the function as in fig 8.

Connection

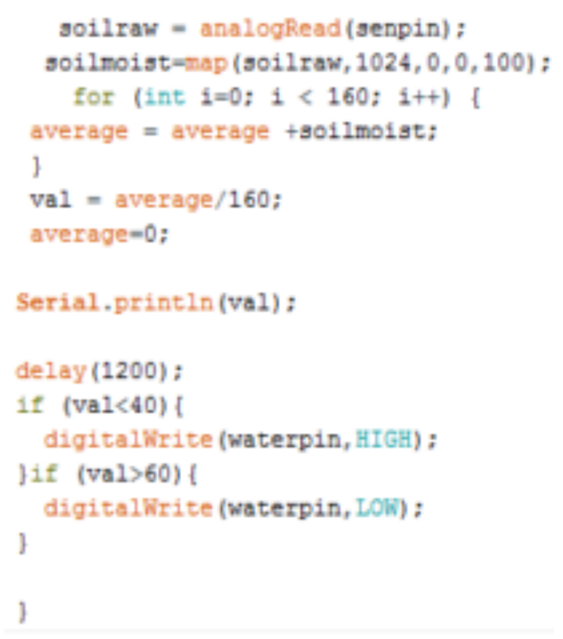
Now connect the components together as described in the table below.

Connect the +ve wire of the 24 volt DC adaptor to the COM pin of the relay and -ve wire of the adaptor to the -ve wire of the valve.



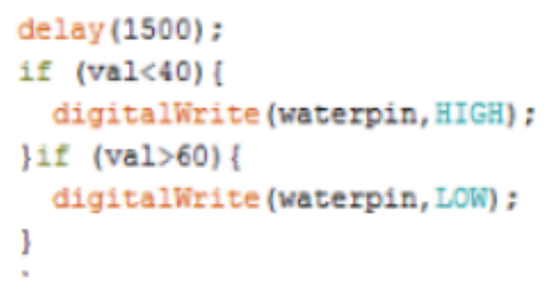
```
void loop(){

      if (Serial.available() > 2) {
  serialA = Serial.parseInt();

  }

    if (serialA== 4){



    digitalWrite(fertilizerpin,1);


    }
     if (serialA== 8){



     digitalWrite(fertilizerpin,0);


    }
     if (serialA== 7){
```

Fig 3.



```
    soilraw = analogRead(senpin);
   soilmoist=map(soilraw,1024,0,0,100);
     for (int i=0; i < 160; i++) {
  average = average +soilmoist;
  }
  val = average/160;
  average=0;

 Serial.println(val);

 delay(1200);
 if (val<40){
   digitalWrite(waterpin,HIGH);
 }if (val>60){
   digitalWrite(waterpin,LOW);
 }


 }
```

Fig 4.



```
delay(1500);
if (val<40){
  digitalWrite(waterpin,HIGH);
}if (val>60){
  digitalWrite(waterpin,LOW);
}
```

fig 5.

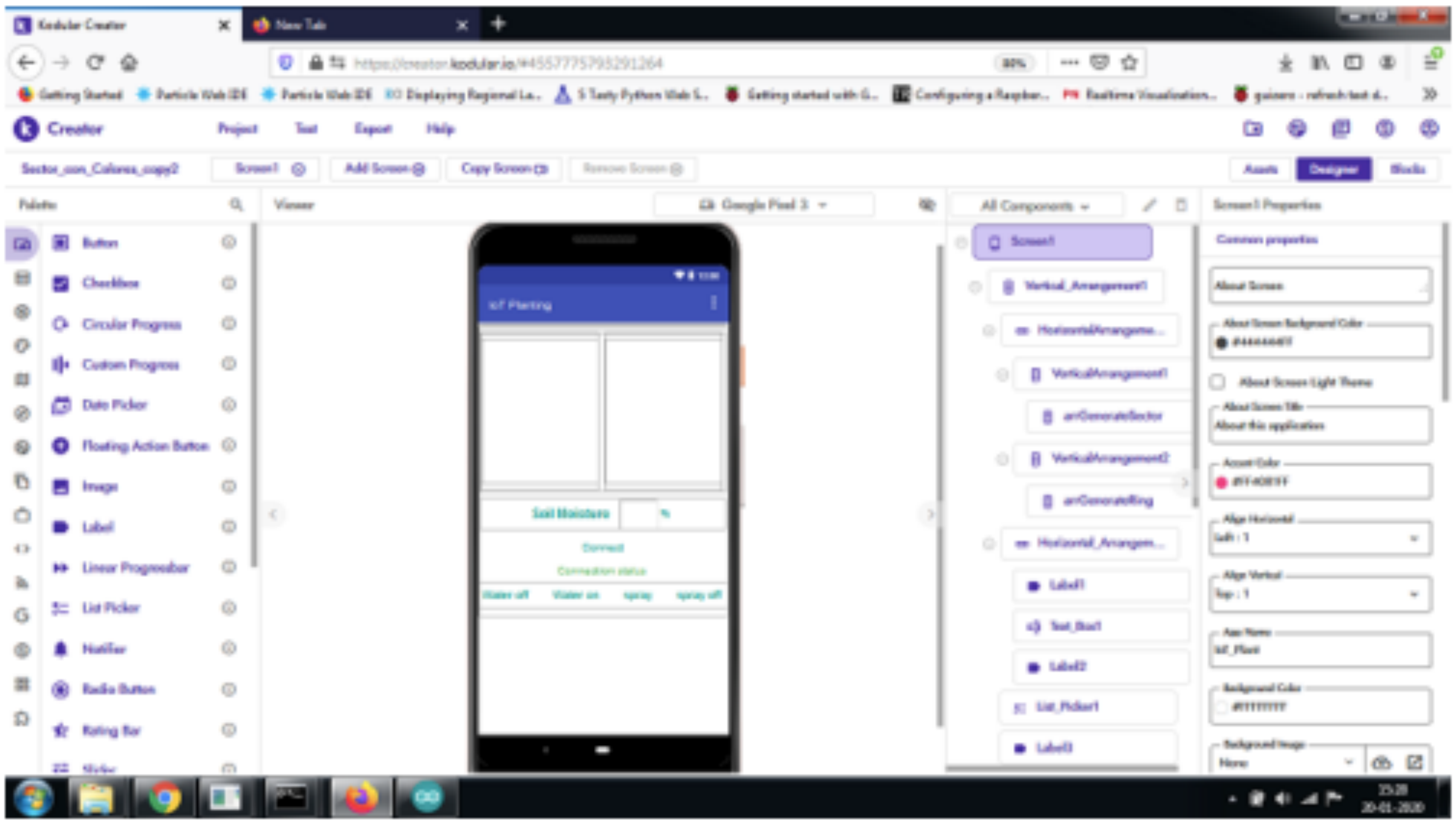


Fig 6. App layout creation

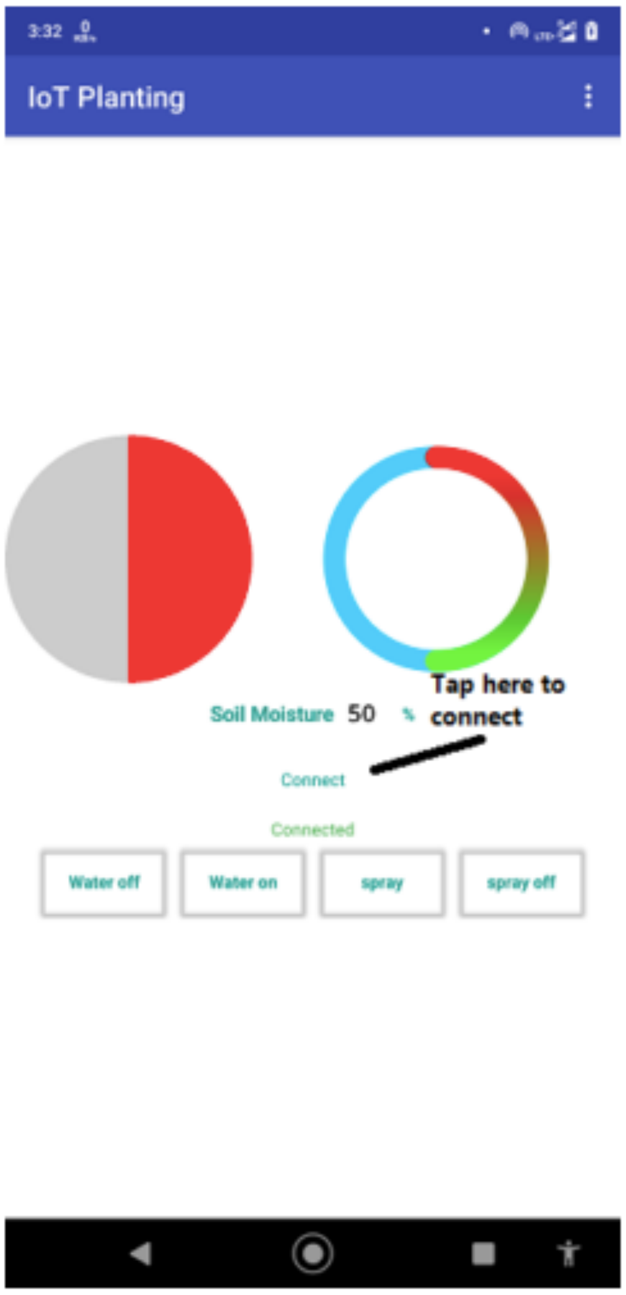


Fig 7.App layout view in phone

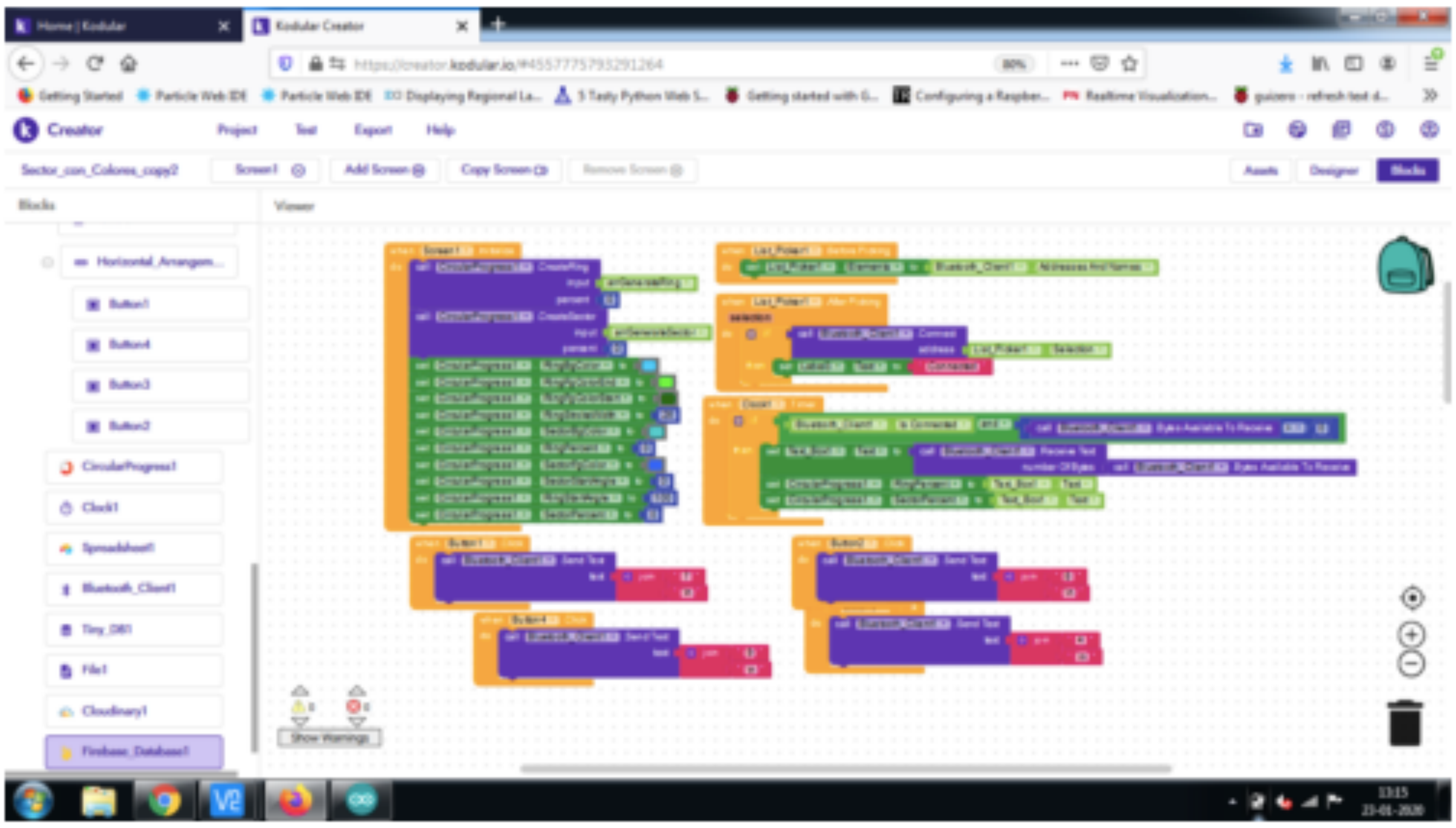


Fig 8.Fig 8. App code blocks

Arduino to soil sensor connection

Arduino	Soil Sensor
5V	VCC
GND	GND
A1 Pin	Analogue OUT

Arduino to Relay

Arduino	Relay
5V	VCC
GND	GND
PIn 6	Relay In 1
PIn 7	Relay In 2

‘

Relay to solenoid valve

Relay	Solenoid valve
Relay NO1	Solenoid valve1 VCC
Relay NO2	Solenoid valve2 VCC(if using fertilizer spray system)

Note: Supply power to the relay module with a separate 5V DC and not with Arduino pin and also check the relay type.

Arduino to Bluetooth HC 05 connection

Arduino	Bluetooth HC 05
5V	VCC
GND	GND
RX	TX
TX	RX

Testing

After connecting the electronic components, cross-check that all of them are connected to the correct power source. Then put the soil sensor and pipe of the solenoid valve into the soil.

Now install the app that we have made and then tap on the connect button (Refer fig 10). You will see a list of Bluetooth devices for connection. Select HC05. (Refer fig 11).

After a successful connection, the app will display the live soil sensor data as in Fig 12. If the soil moisture is below 50 per cent, then it means that the soil is dry and needs water, which will be automatically provided to it. After getting a sufficient amount, the supply will turn off. You can also spray fertilizers or pesticides using the app by tapping the spray on/off button. For that, you need to make an additional setup with the pipe to spray pesticides and fertilizers.

Troubleshooting

You might come across some errors while using the app (due to version or Bluetooth connectivity) as shown in Fig 13, 14. To get rid of them, simply go back or tap anywhere outside the error message box.

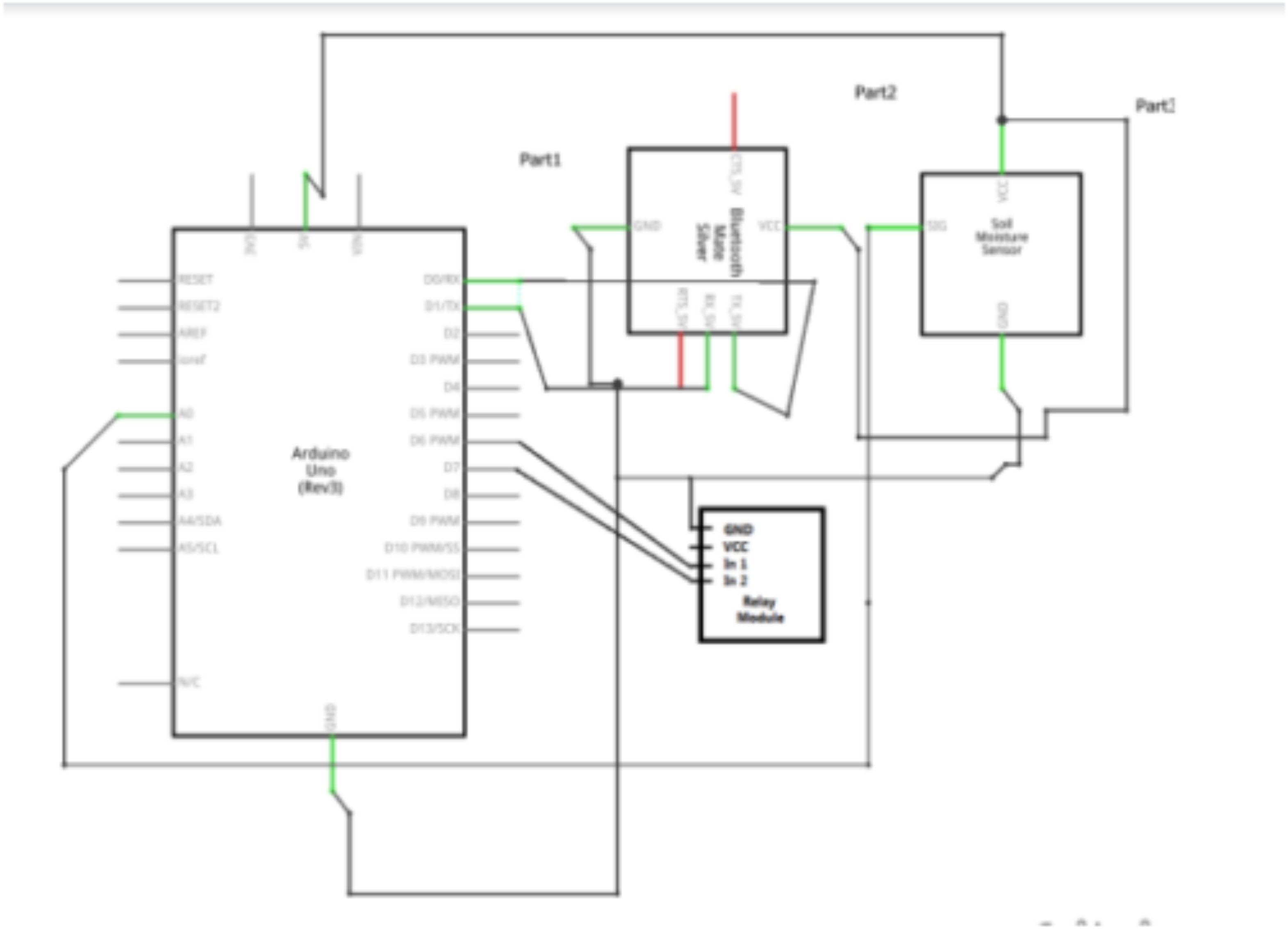
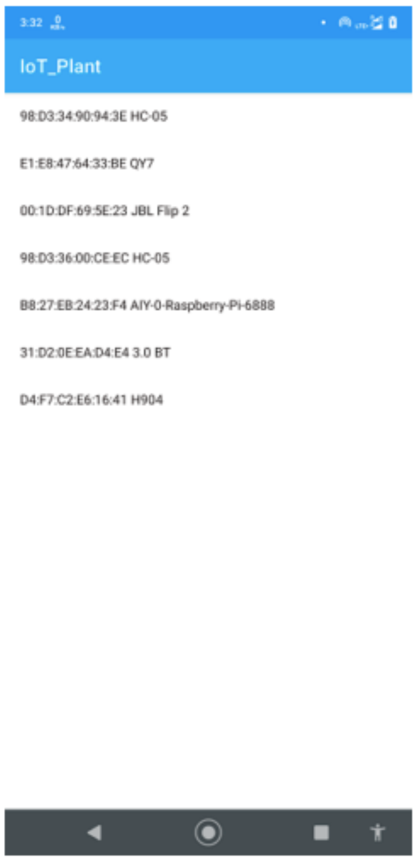


Fig 9.Connection

Download Source Code

You can watch Video for this DIY here: https://youtu.be/NtWF_Tx_BNo



```
98:D3:34:90:94:3E HC-05
E1:E8:47:64:33:BE QY7
00:1D:DF:69:5E:23 JBL Flip 2
98:D3:36:00:CE:EC HC-05
B8:27:EB:24:23:F4 AIY-0-Raspberry-Pi-6888
31:D2:0E:EA:D4:E4 3.0 BT
D4:F7:C2:E6:16:41 H904
```

Fig 11.

```
98:D3:34:90:94:3E HC-05
E1:E8:47:64:33:BE QY7
00:1D:DF:69:5E:23 JBL Flip 2
98:D3:36:00:CE:EC HC-05
B8:27:EB:24:23:F4 AIY-0-Raspberry-Pi-6888
31:D2:0E:EA:D4:E4 3.0 BT
D4:F7:C2:E6:16:41 H904
```

Fig 11.

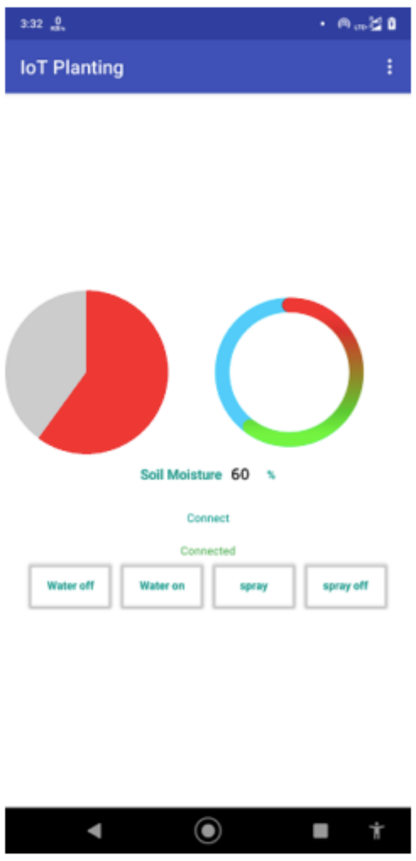
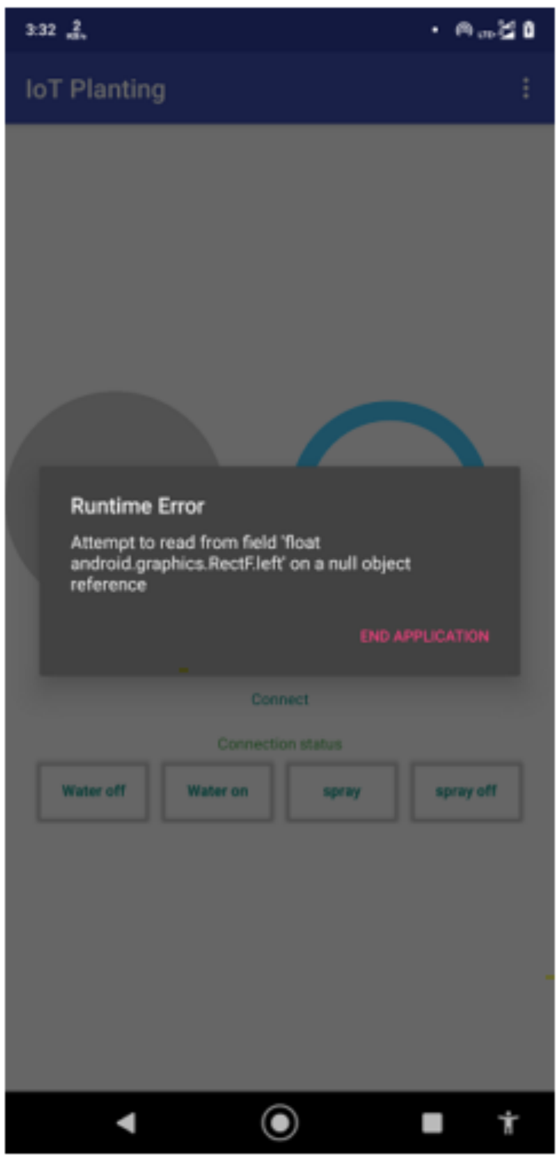


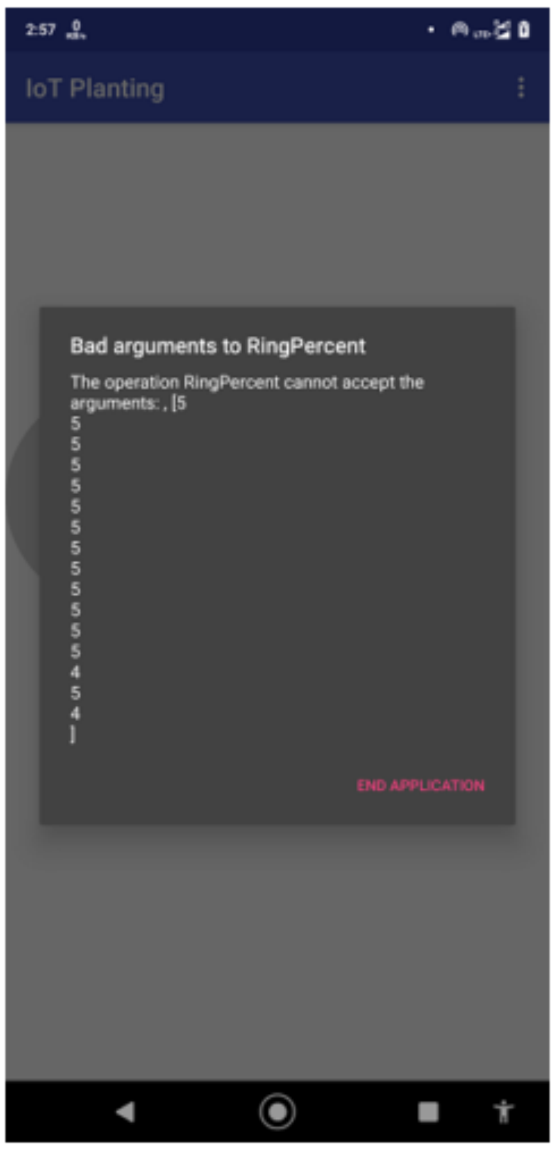
Fig 12. Showing the data.



Runtime Error

Attempt to read from field 'float android.graphics.RectF.left' on a null object reference

Fig 13. Version erro



Bad arguments to RingPercent

```
The operation RingPercent cannot accept the
arguments: , [5
5
5
5
5
5
5
5
5
5
5
5
5
4
5
4
]
```

Fig 14. Bluetooth connectivity error

24

12V BATTERY ABSORB AND FLOAT CHARGER

BY FAYAZ HASSAN

Most battery chargers stop charging the battery when it attains its maximum charging voltage set by the circuit. This 12V battery charger circuit charges the battery at a particular voltage, that is, absorption voltage, and once the maximum charging voltage is attained, the charger changes the output voltage to float voltage for maintaining the battery at that voltage. Absorption and floating voltages are dependent on the type of battery.

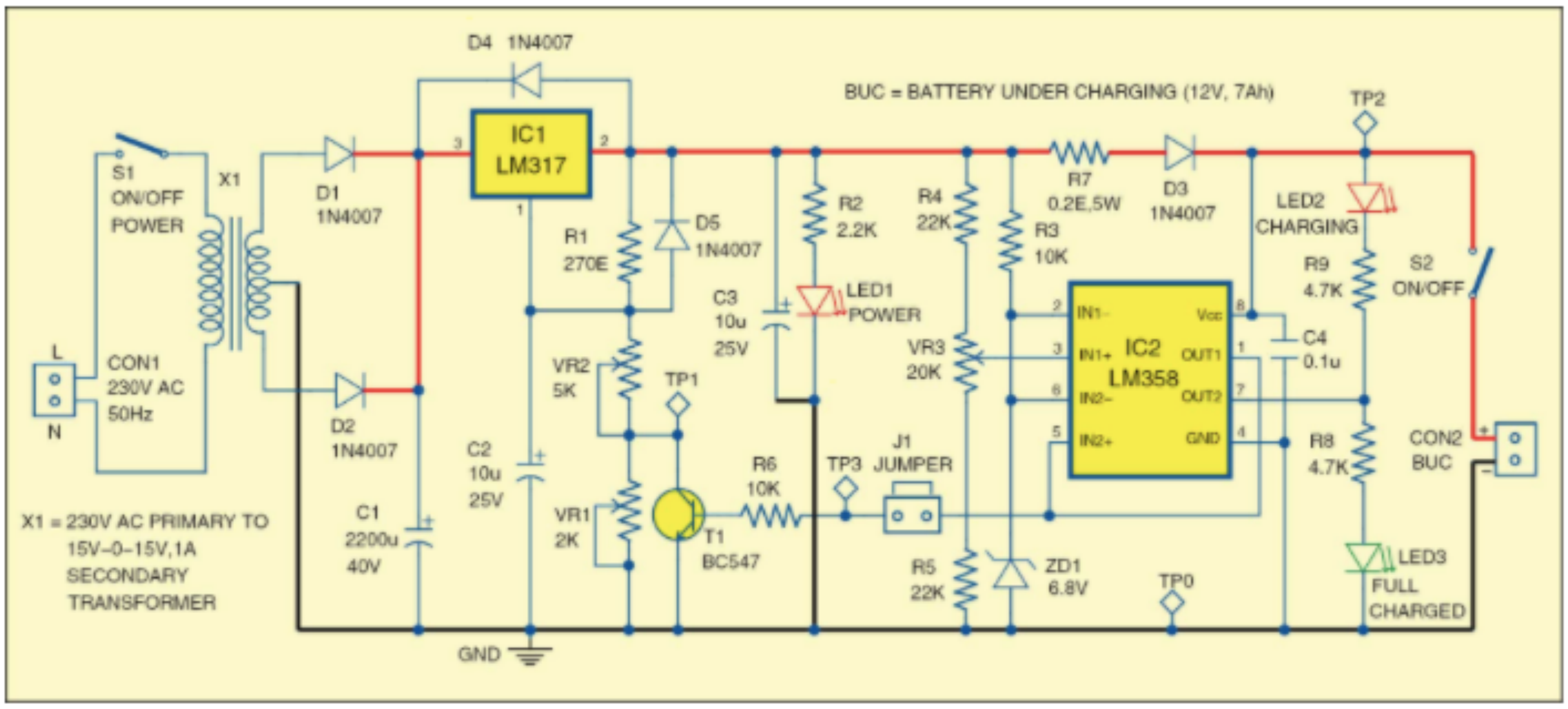


Fig1: 12V battery charger circuit

For this charger, voltages are set for a sealed lead-acid (SLA) 12V, 7Ah battery, for which absorption voltage is 14.1V to 14.3V and floating voltage is 13.6V to 13.8V. For safe working and to avoid overcharging of battery, absorption voltage is selected as 14.1V and floating voltage is selected as 13.6V. These values are to be set as specified by the battery manufacturer.

12V battery charger circuit

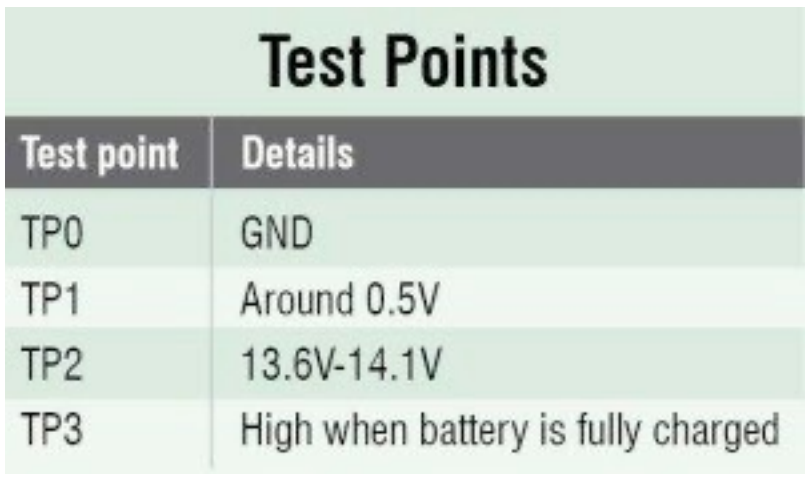
Circuit diagram of the 12V battery absorb and float charger is shown in Fig. 1. It is built around step-down transformer X1, adjustable voltage regulator LM317 (IC1), op-amp comparator LM358 (IC2) and a few other components. The 230V AC primary to 15V-0-15V, 1A secondary transformer used in this circuit steps down mains voltage, which is rectified by diodes D1 and D2 and smoothened by capacitor C1. This voltage is given to the input of LM317 for regulation.

The basic circuit is a regulated power

PARTS LIST

Semiconductors:	
IC1	- LM317 adjustable voltage regulator
IC2	- LM358 op-amp
T1	- BC547 npn transistor
LED1-LED3	- 5mm LED
ZD1	- 6.8V zener diode
D1-D5	- 1N4007 rectifier diode
Resistors (all 1/4-watt, ±5% carbon, unless stated otherwise):	
R1	- 270-ohm
R2	- 2.2-kilo-ohm
R3, R6	- 10-kilo-ohm
R4, R5	- 22-kilo-ohm
R7	- 0.2-ohm, 5W
R8, R9	- 4.7-kilo-ohm
VR1	- 2-kilo-ohm potmeter
VR2	- 5-kilo-ohm potmeter
VR3	- 20-kilo-ohm potmeter
Capacitors:	
C1	- 2200µF, 40V electrolytic
C2, C3	- 10µF, 25V electrolytic
C4	- 0.1µF ceramic disk
Miscellaneous:	
X1	- 230V AC primary to 15V-0-15V, 1A secondary transformer
CON1, CON2	- 2-pin connector terminal
	- 12V, 7Ah rechargeable battery
J1	- 2-pin connector for jumper J1
S1, S2	- On/off switch
	- Heat sink for LM317

supply using LM317, with a control on output by changing resistance at adjust pin 1. A good heat-sink is required for LM317. LM358 is a dual-operation amplifier that is used here to control overcharging of the battery. Capacitor C4 should be as near as possible to pin 1 of IC2. Jumper J1 is used for calibration (set-up). While setting the charging voltage, remove the jumper and connect it back after calibration.


Test Points

Test point	Details
TP0	GND
TP1	Around 0.5V
TP2	13.6V-14.1V
TP3	High when battery is fully charged

For initial setup, remove jumper J1, switch off S2, switch on S1 and adjust potmeter VR2 to get 13.6V at test point TP2. Adjust potmeter VR3, so that LED2 begins to glow. Adjust potmeter VR1 to read 0.5V (difference of 14.1V and 13.6V) at test point TP1. Adjust VR2 to read 14.1V at test point TP2.

With these settings, TP2 should read 14.1V when there is low voltage at test point TP3, and 13.6V when there is high voltage at test point TP3. Connect jumper J1. The charger is now ready for use. Connect the 12V battery under charging (BUC), with correct polarity, at CON2. Switch on S2; one of the LEDs out of LED2 and LED3 will light up (most likely it would be LED2). If neither of these light up, check the connections; battery could be dead. Switch on S1 for charging. Fully charged status of the battery will be indicated by glowing of LED3.

Do not worry if you forget to switch off the charger. The charger is on floating voltage (13.6V) now and it can be kept in this charging mode forever.

Construction and testing

A single-side PCB for the 12V battery absorb and float charger circuit is shown in Fig. 2 and its component layout in Fig. 3. Assemble the circuit on the PCB, except transformer X1 and the battery under charge (BUC).

Enclose the PCB in a small box. Fix the battery terminal on the front of the box for connecting the BUC. Connect switches S1 and S2, potmeters VR1 through VR3, etc on the body of the box.

EFY notes

1. Switch off S2 or disconnect battery terminals to avoid unnecessary discharge of battery when not charging, that is, when S1 is switched off.
2. Connect the battery with correct polarity.
3. Casing of IC1 should not be connected to ground, so use insulation.

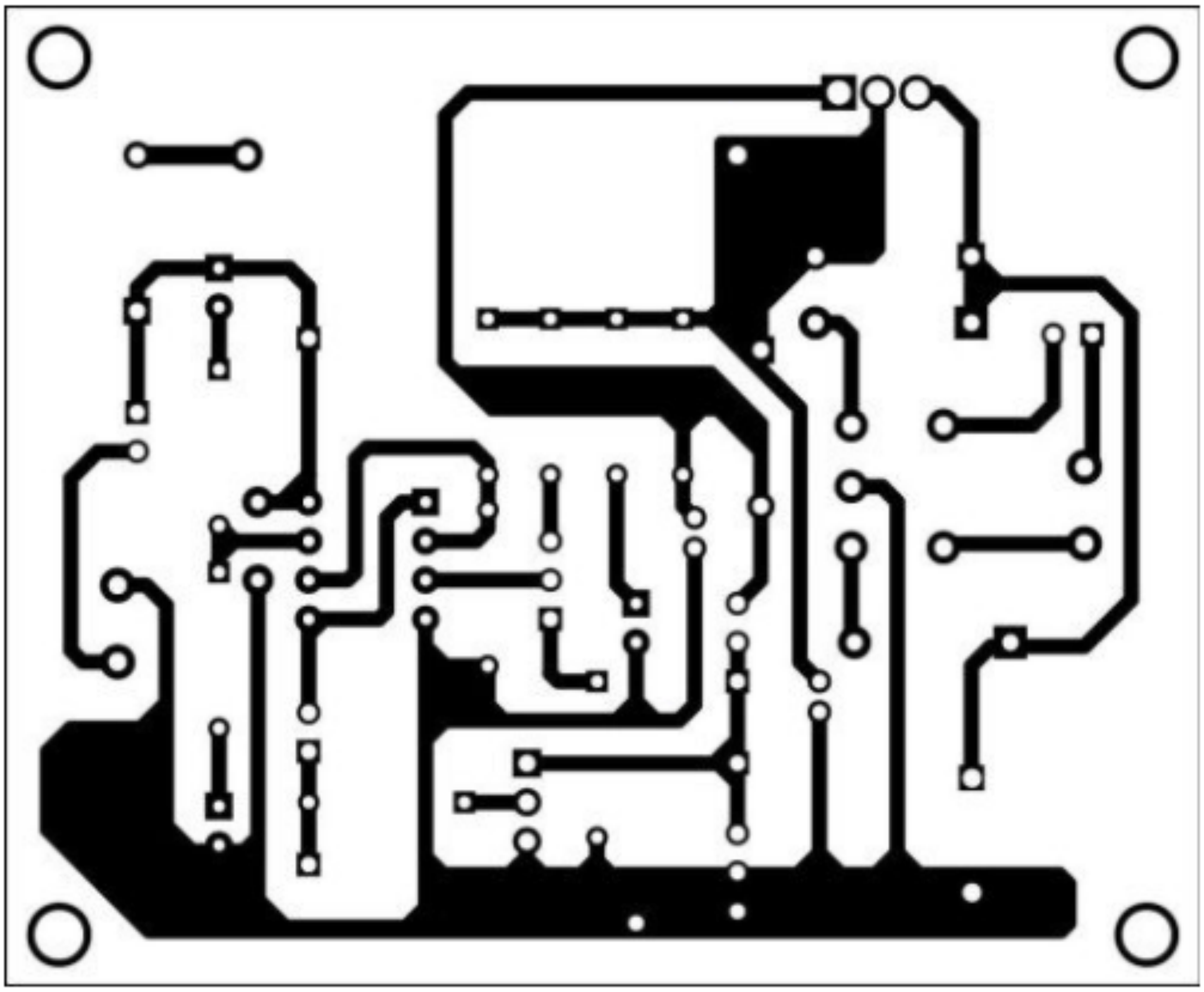


Fig. 2: PCB of the 12V battery charger

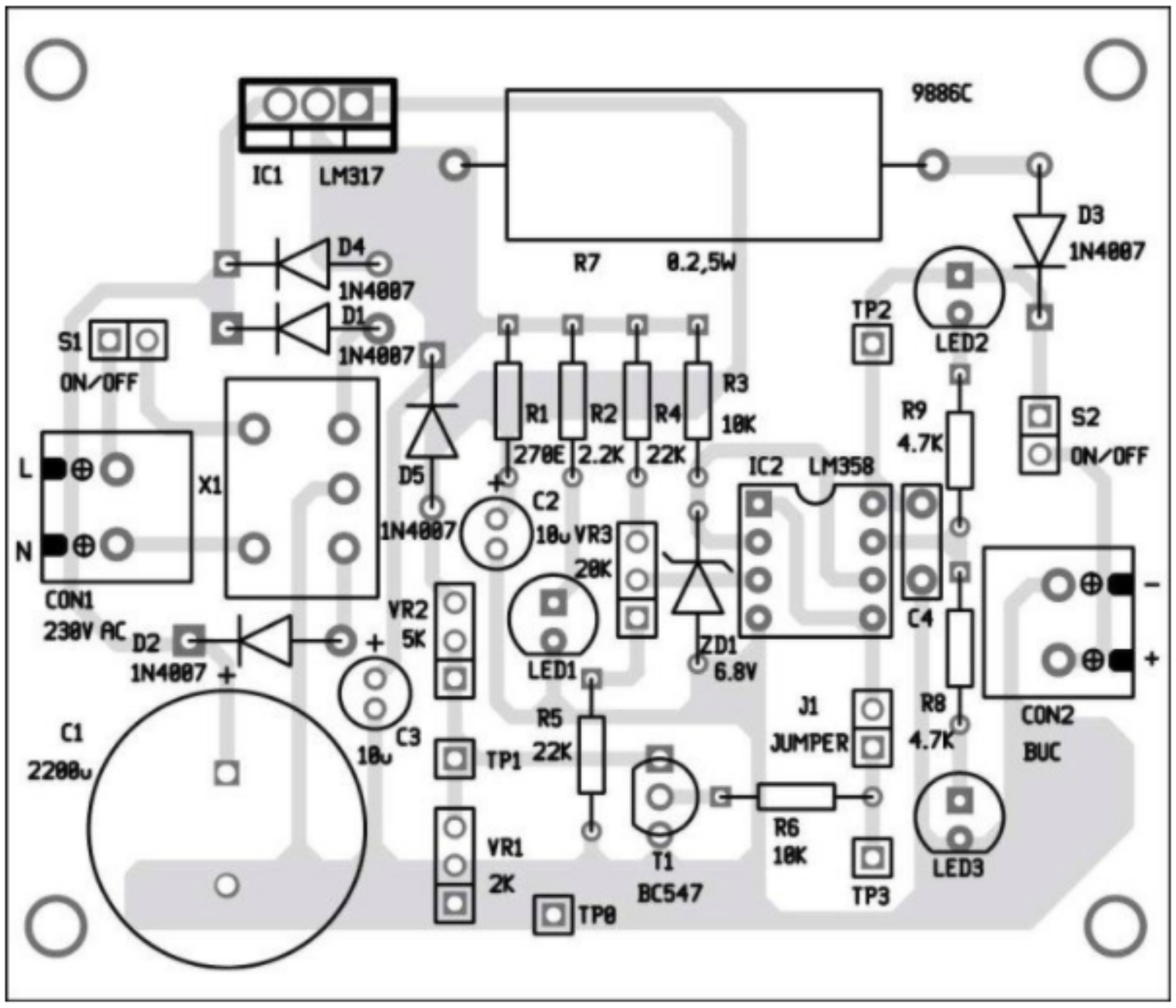


Fig. 3: Component layout of the PCB

Download the PCB and component layout PDFs: click here

25

SMART WEARABLE DEVICE FOR DISTRESS CALLS

BY ASHWINI KUMAR SINHA

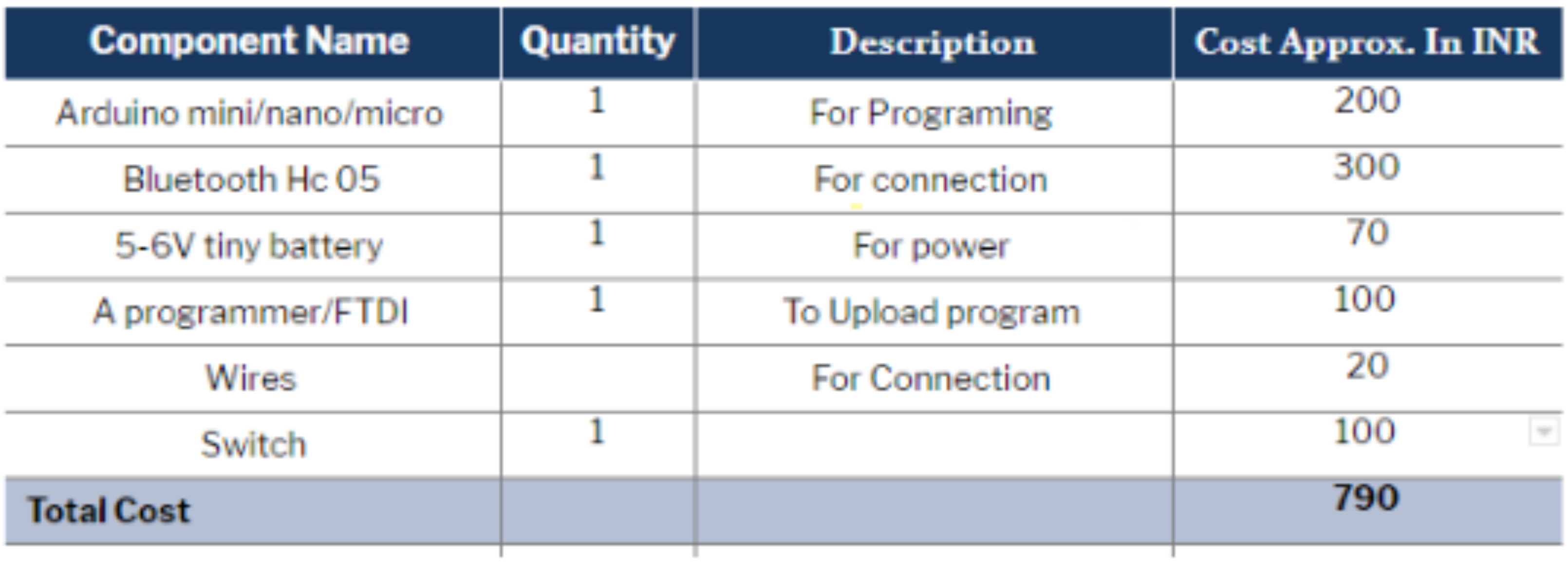
We all know how it feels to be stuck in a helpless situation and the stress it causes. Imagine being encountered by a dangerous person trying to harm you and there is no way for you to call for help. The stress faced here would lead to increased panic.

Worry not. Today we are going to make a prototype of a smart wearable device for distress calls that can connect to your phone via a smart app. At a simple press of button on the smart device, a help message with your current location will be automatically sent to your family member or trusted person who can then come to your rescue.

This device is especially helpful for women to make them feel safe while venturing out.

So, let’s get started with the project.

Materials Required

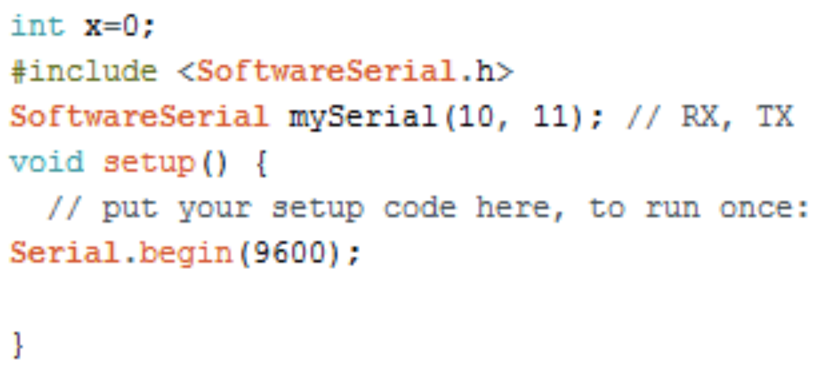


Component Name	Quantity	Description	Cost Approx. In INR
Arduino mini/nano/micro	1	For Programing	200
Bluetooth Hc 05	1	For connection	300
5-6V tiny battery	1	For power	70
A programmer/FTDI	1	To Upload program	100
Wires		For Connection	20
Switch	1		100
Total Cost			**790**

We will require the following basic electronic components.

Coding

Initialise the SoftwareSerial function with a baud rate of 9600 for Bluetooth communication with the smart app. Then declare an integer variable for storing the value of switch input. (Refer Fig 1.)



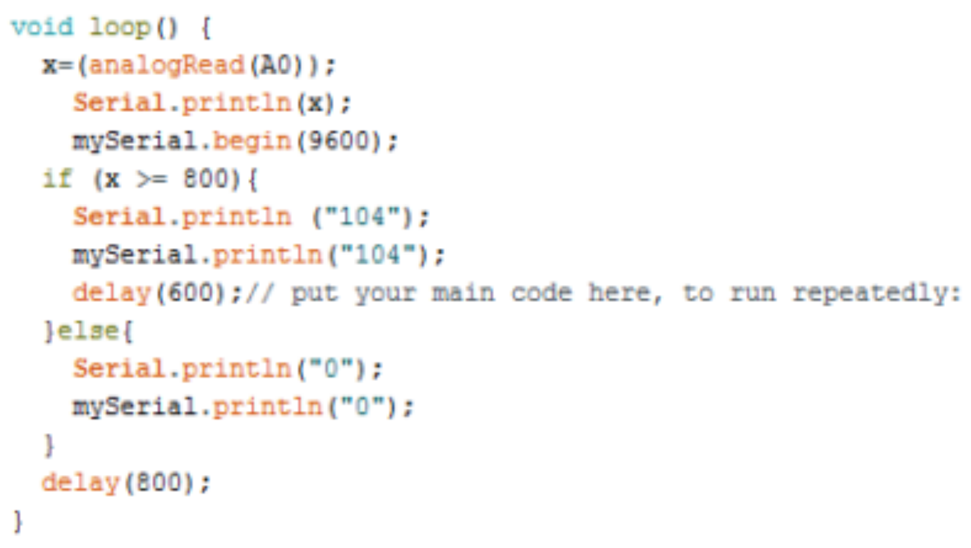
```
int x=0;
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // RX, TX
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600);

}
```

Fig 1. Arduino code

Next, create a loop function for checking the previously stored switch value at analog pin A0. We also create an ‘if’ condition for sending a unique number code to the smart app whenever the switch on the device is pressed and the condition holds true. (Refer Fig 2.)

Creating the smart app

After writing the Arduino code, let's now make the smart app. Go to www.kodular.io, click Create Apps and create your account (sign in if you already have one). It's free and simple for creating Android apps. Once logged in, select Create project and give it a name. Then from the list of tools given on the left hand-side of the screen (Palette), select and add the following tools for the UI display. Drag-and-drop them into the virtual phone screen. (Refer Fig 3.)



```
void loop() {
  x=(analogRead(A0));
    Serial.println(x);
    mySerial.begin(9600);
  if (x >= 800){
    Serial.println ("104");
    mySerial.println("104");
    delay(600);// put your main code here, to run repeatedly:
  }else{
    Serial.println("0");
    mySerial.println("0");
  }
  delay(800);
}
```

Fig 2. Arduino code.

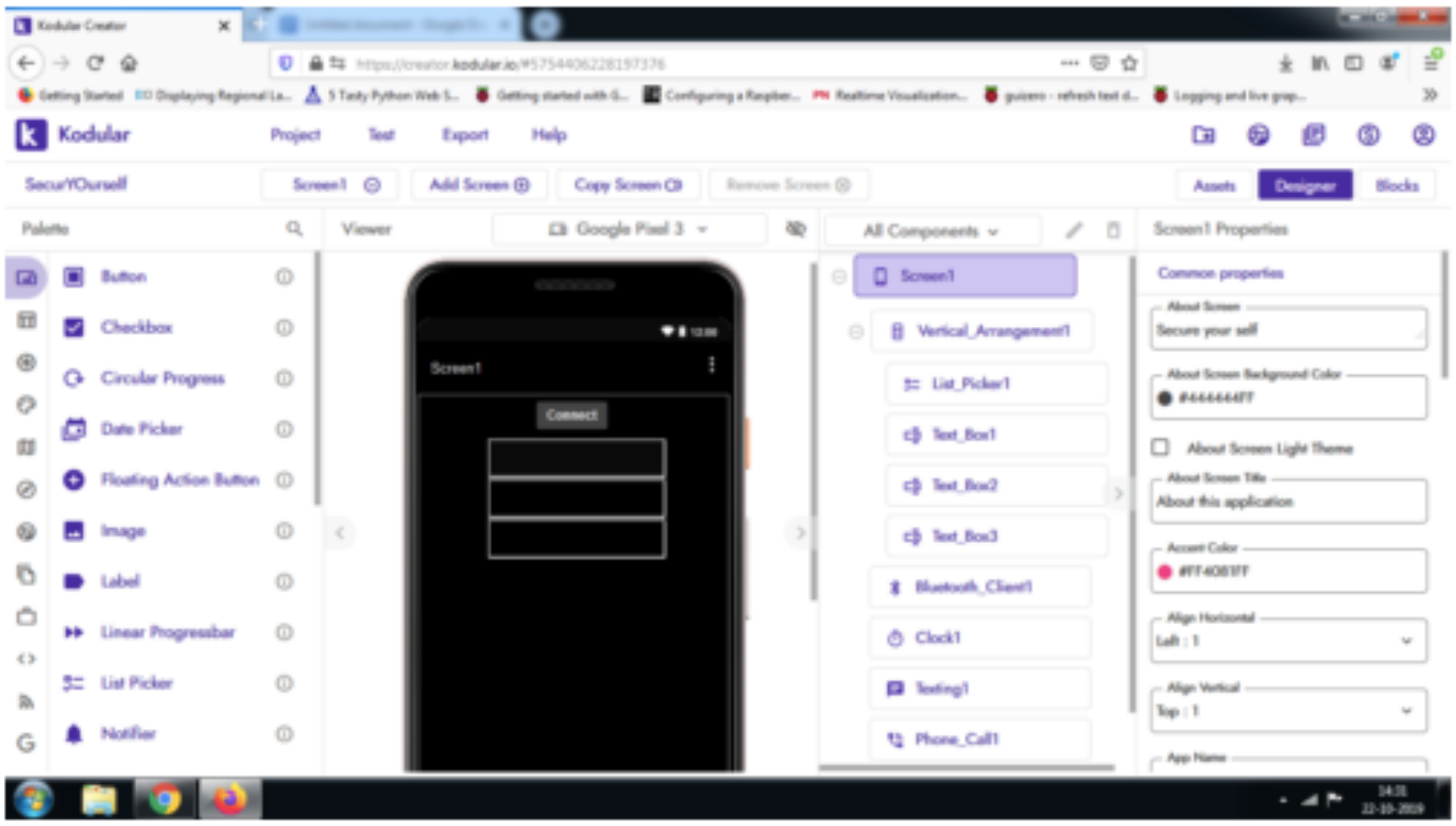


Fig 3. Creating the app

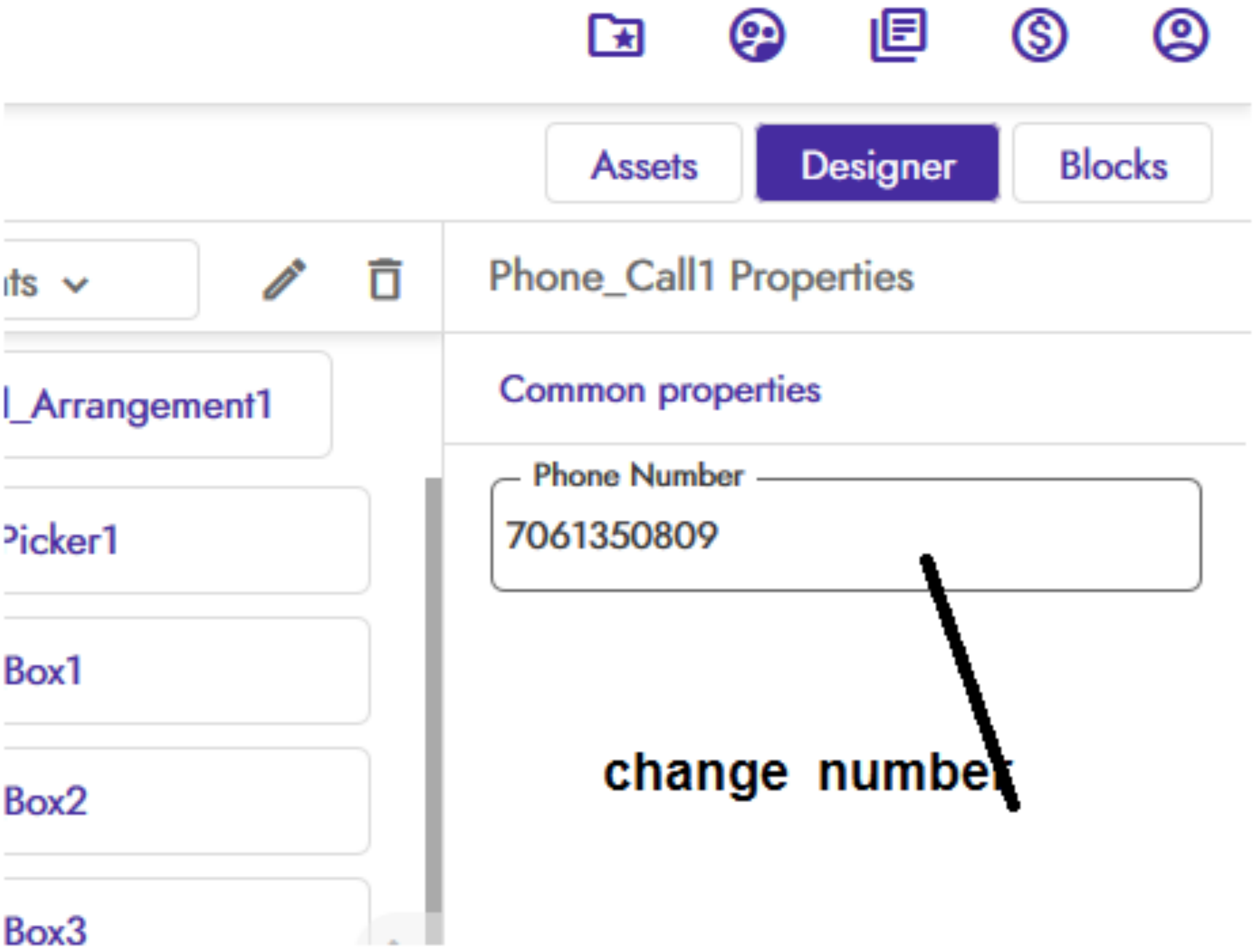
Tools

- 3 Text_Boxes
- 1 Bluetooth_Client (select Connectivity ? Bluetooth Client)
- 1 Clock (select Sensors ? Clock)

- 1 Texting (select Social ? Texting)
- 1 Phone_Call (select Social ? Phone Call)
- 1 Location_Sensor (select Sensors ? Sensor Location)
- 1 List_Picker (select User Interface ? List Picker)
- 1 Vertical_Arrangement (optional)

Note: The above-mentioned tools are named as components in the software, so don't get confused.

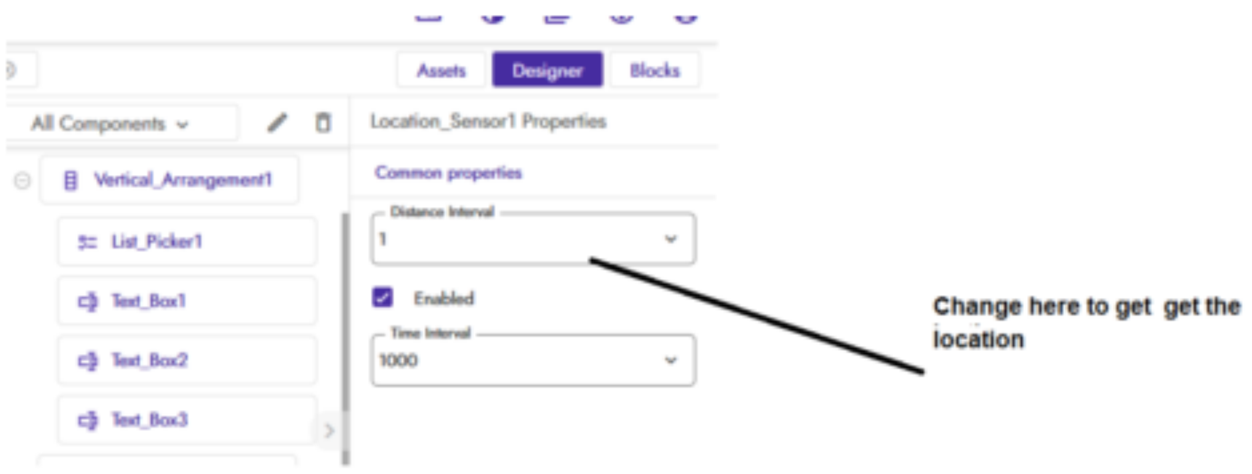
After adding the necessary tools for the UI display, click on Phone_Call and enter the phone number you wish to contact. Then click on Texting and enter the desired message along with the same phone number. (Refer Fig 4.)



Phone Number: 7061350809

Fig 4. Phone number

Then click on Location_Sensor to set Distance Interval to 1 and Time Interval to 1000. These numbers indicate the threshold values for time and location change with every step the user takes. (Refer Fig 5.)



Location_Sensor1 Properties

Common properties

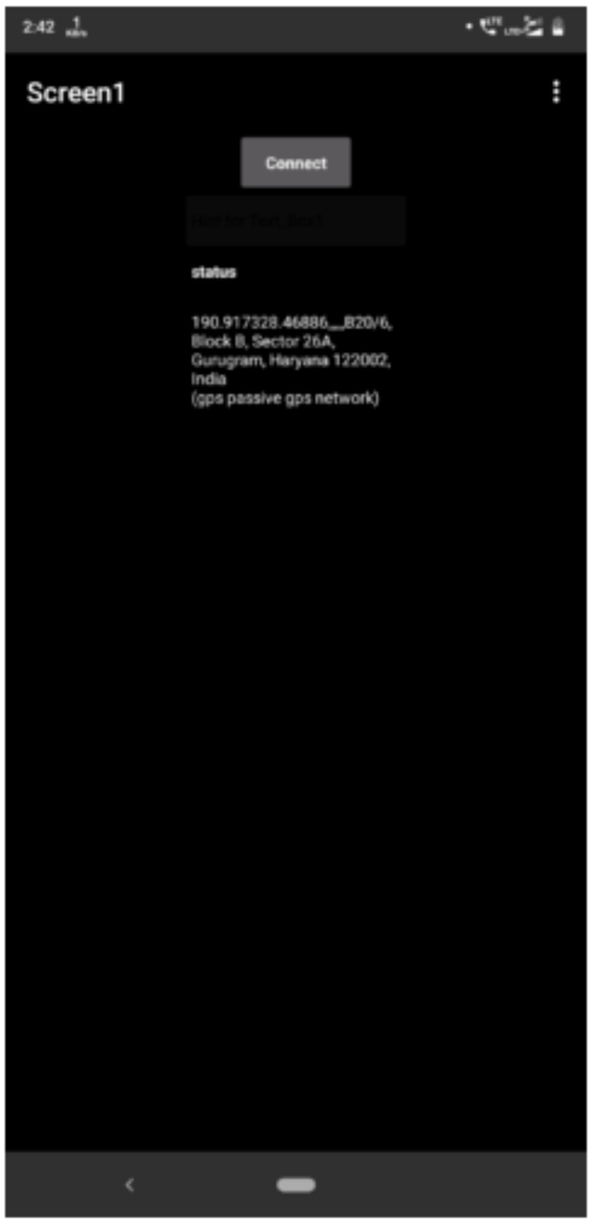
Distance Interval: 1

☑ Enabled

Time Interval: 1000

Fig 5. Time and Location values

The final app layout is as shown below. (Refer Fig 6.)



status

190.917328.46886,_B20/6,
Block B, Sector 26A,
Gurugram, Haryana 122002,
India
(gps passive gps network)

Fig 6. App layout.

Programming the smart app

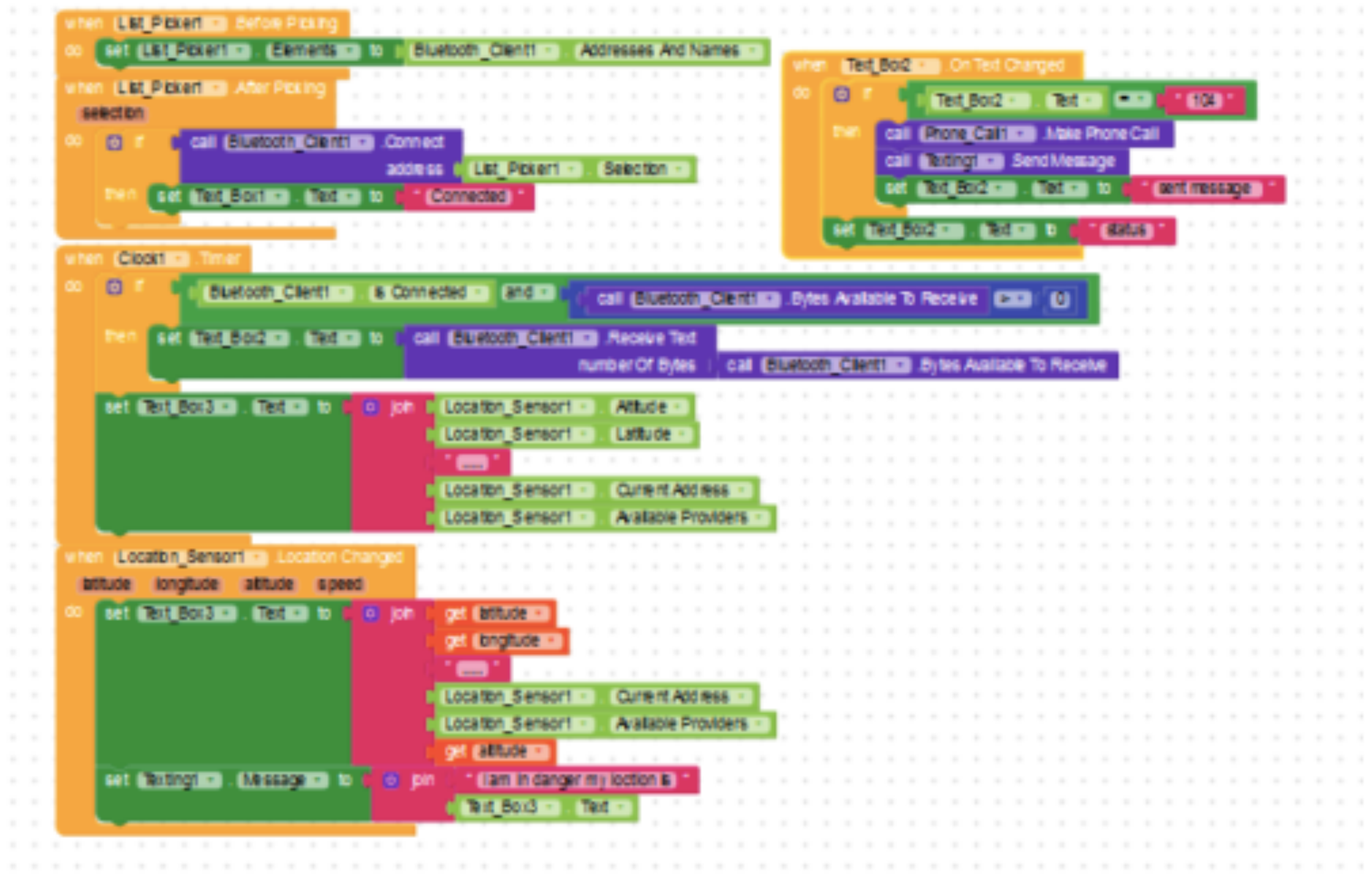
Now go to the top-right corner on the same window and select Blocks. You will see the layout for coding. Program the smart app as shown below. (Refer Fig 7.)

Download Source Code

Connection

Connect the components on the Arduino board as described in Fig. 8.

Arduino PIN 10 --------------HC 05 RX
Arduino PIN 11---------------HC 05 TX
Arduino Pin GND-------------HC 05 GND
Arduino Pin VCC-------------HC 05 5V
Arduino A0 -----------------Switch Pin



```
when List_Picker1 .Before Picking
do  set List_Picker1 . Elements to  Bluetooth_Client1 . Addresses And Names

when List_Picker1 .After Picking
  selection
do  if   call Bluetooth_Client1 .Connect
                              address  List_Picker1 . Selection
    then set Text_Box1 . Text to " Connected "

when Clock1 .Timer
do  if   Bluetooth_Client1 . is Connected  and  call Bluetooth_Client1 .Bytes Available To Receive  >  0
    then set Text_Box2 . Text to  call Bluetooth_Client1 .Receive Text
                                      numberOf Bytes  call Bluetooth_Client1 .Bytes Available To Receive
    set Text_Box3 . Text to  join  Location_Sensor1 . Altitude
                                   Location_Sensor1 . Latitude
                                   " ..... "
                                   Location_Sensor1 . Current Address
                                   Location_Sensor1 . Available Providers

when Location_Sensor1 .Location Changed
  latitude  longitude  altitude  speed
do  set Text_Box3 . Text to  join  get latitude
                                   get longitude
                                   " ..... "
                                   Location_Sensor1 . Current Address
                                   Location_Sensor1 . Available Providers
                                   get altitude
    set Texting1 . Message to  join  " I am in danger my loction is "
                                     Text_Box3 . Text

when Text_Box2 .On Text Changed
do  if   Text_Box2 . Text  =  " 104 "
    then call Phone_Call1 .Make Phone Call
         call Texting1 .Send Message
         set Text_Box2 . Text to " sent message "
    set Text_Box2 . Text to " status "
```

Fig 7. Kodular code blocks

Testing

After making all the proper connections, power the device with a 5V battery and then connect the smart app with Bluetooth. On pressing the switch, the smart app will automatically send the message alongwith the location to the set phone number.

You can watch Video for this DIY here: https://youtu.be/ygXLw5HOWRU

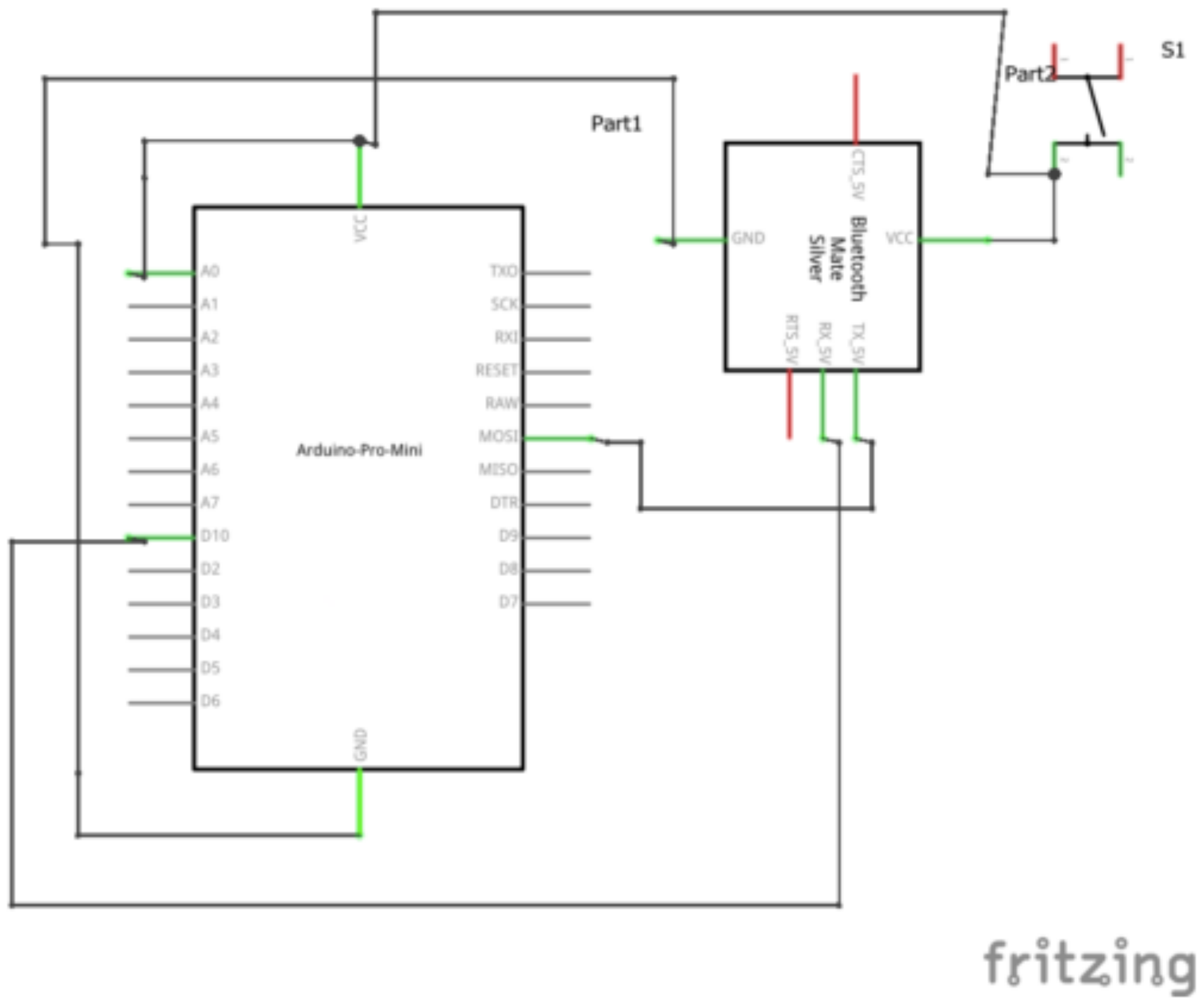


Fig 8. Connection diagram.

26

GSM-BASED CELLULAR IOT HOME AUTOMATION

BY ASHWINI KUMAR SINHA

What if you are out of your home (and/or in another country) and suddenly realise that you have forgotten to turn off the geyser, lights and other electrical appliances? This costly mistake can increase the electricity bill as well as put your and others lives in danger.

Fear not. We are going to make an IoT-based cellular device for home automation, which will help you in controlling your home appliances, no matter where you are.

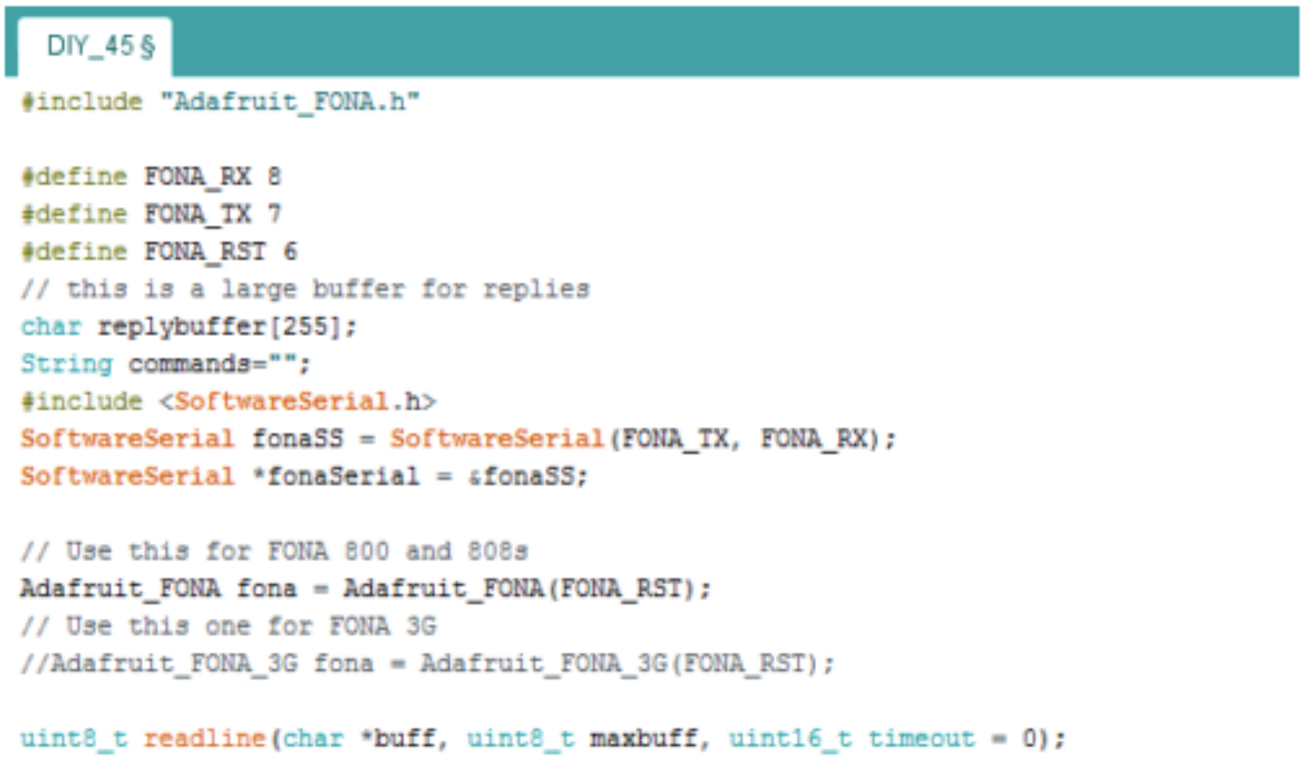
Bill Of Material

Component Name	Quantity	Description	Cost Approx. In INR
Arduino Uno/ Mini	1	For Programming	200
SIM 800L	1	GSM Module	300
Relay	1	2 Channel Relay	200
Wires	30cm	For Connection	30
Total Cost			**730**

Coding

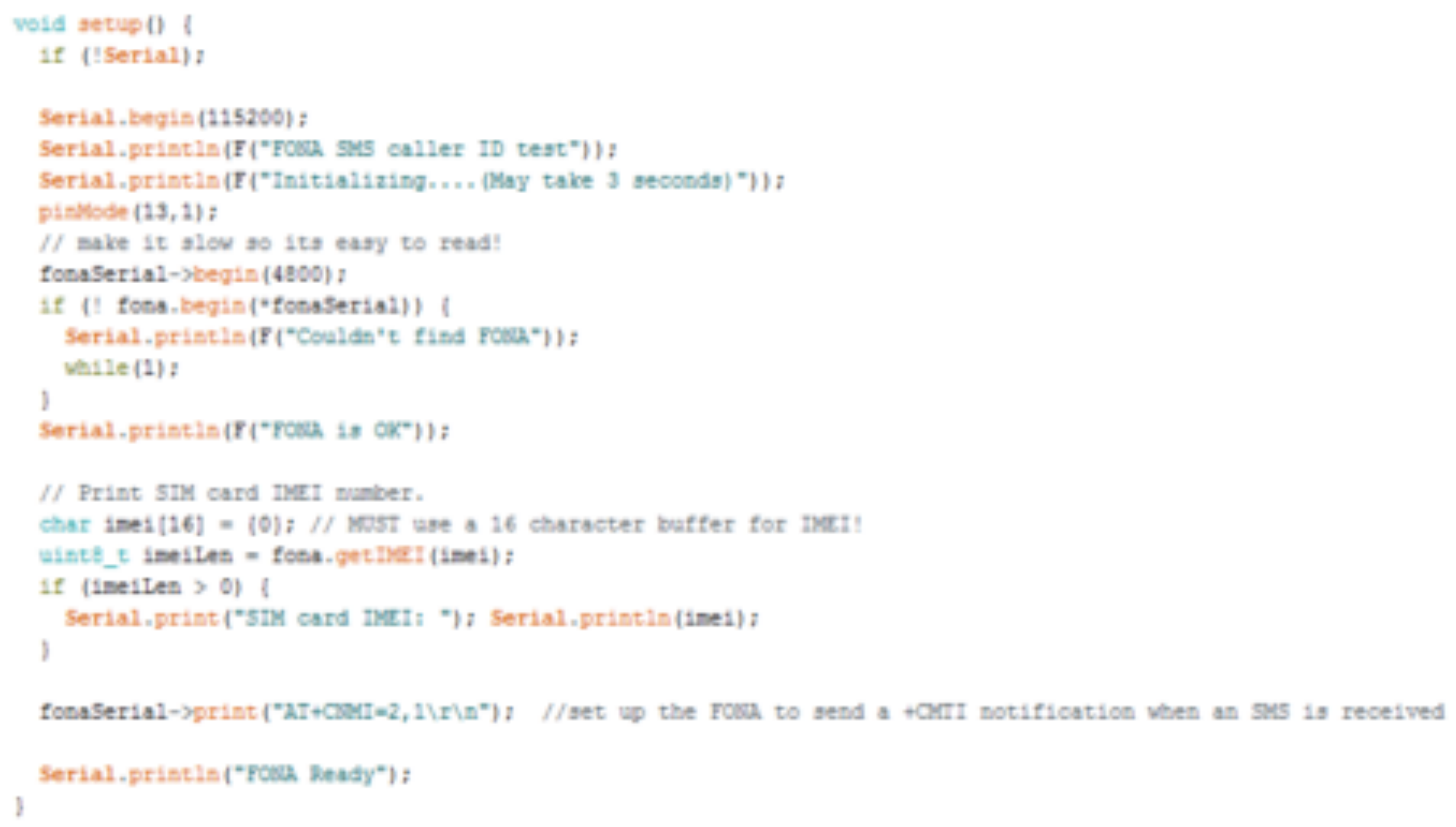
Install the Adafruit Fona library in Arduino IDE. Inside this library, set the serial and reset pins for the SIM 800L module, and after that, set its baud rate to 4800 in the setup function.

Now, we will create a loop where we check the incoming message and with the help of the if condition, we will set a function for the command to be sent from message. (Refer Fig 2,3,4,5).



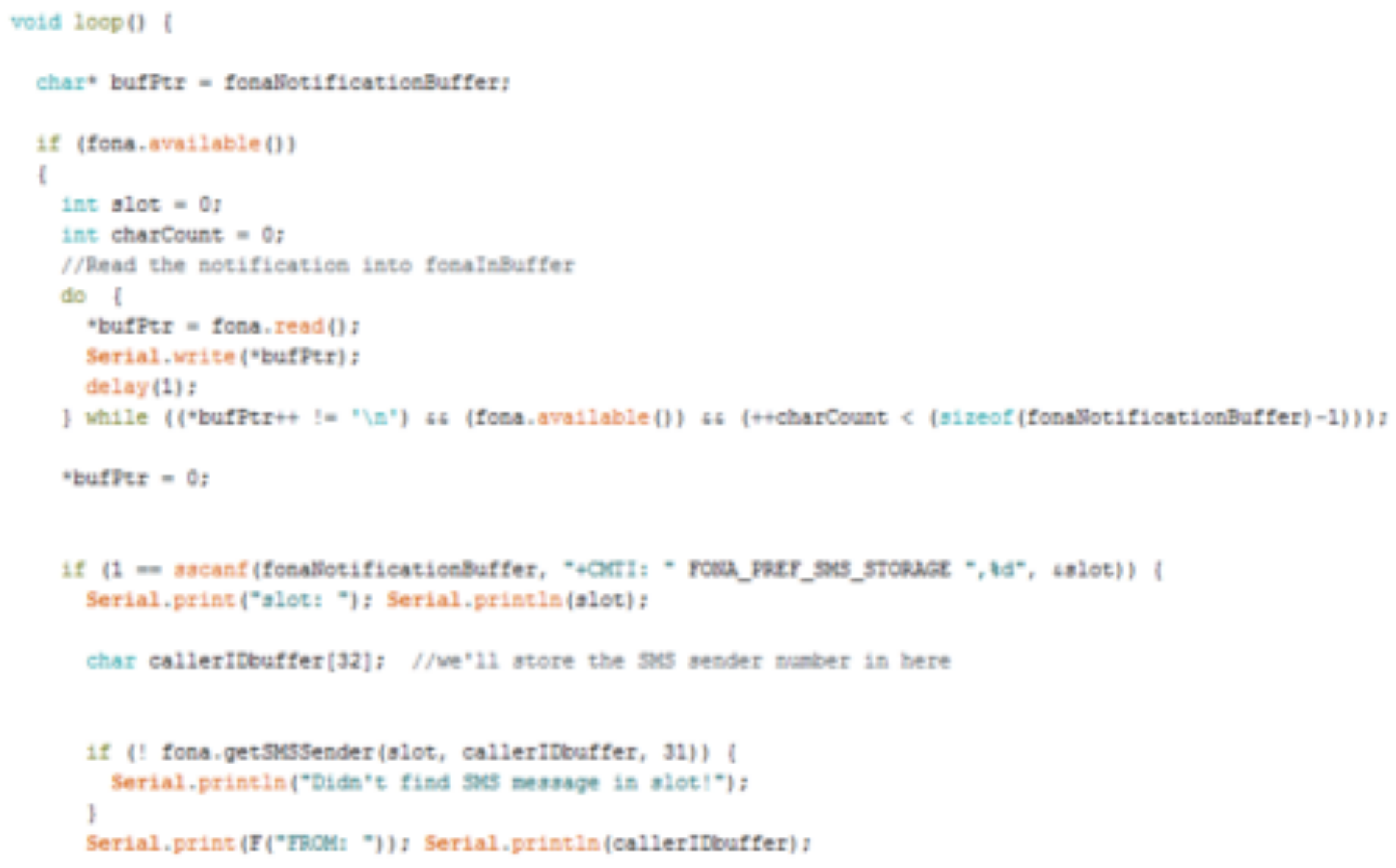
```
#include "Adafruit_FONA.h"

#define FONA_RX 8
#define FONA_TX 7
#define FONA_RST 6
// this is a large buffer for replies
char replybuffer[255];
String commands="";
#include <SoftwareSerial.h>
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

// Use this for FONA 800 and 808s
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);
// Use this one for FONA 3G
//Adafruit_FONA_3G fona = Adafruit_FONA_3G(FONA_RST);

uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);
```

Fig 2.



```
void setup() {
  if (!Serial);

  Serial.begin(115200);
  Serial.println(F("FONA SMS caller ID test"));
  Serial.println(F("Initializing....(May take 3 seconds)"));
  pinMode(13,1);
  // make it slow so its easy to read!
  fonaSerial->begin(4800);
  if (! fona.begin(*fonaSerial)) {
    Serial.println(F("Couldn't find FONA"));
    while(1);
  }
  Serial.println(F("FONA is OK"));

  // Print SIM card IMEI number.
  char imei[16] = {0}; // MUST use a 16 character buffer for IMEI!
  uint8_t imeiLen = fona.getIMEI(imei);
  if (imeiLen > 0) {
    Serial.print("SIM card IMEI: "); Serial.println(imei);
  }

  fonaSerial->print("AT+CNMI=2,1\r\n");  //set up the FONA to send a +CMTI notification when an SMS is received

  Serial.println("FONA Ready");
}
```

Fig 3.



```
void loop() {

  char* bufPtr = fonaNotificationBuffer;

  if (fona.available())
  {
    int slot = 0;
    int charCount = 0;
    //Read the notification into fonaInBuffer
    do  {
      *bufPtr = fona.read();
      Serial.write(*bufPtr);
      delay(1);
    } while ((*bufPtr++ != '\n') && (fona.available()) && (++charCount < (sizeof(fonaNotificationBuffer)-1)));

    *bufPtr = 0;


    if (1 == sscanf(fonaNotificationBuffer, "+CMTI: " FONA_PREF_SMS_STORAGE ",%d", &slot)) {
      Serial.print("slot: "); Serial.println(slot);

      char callerIDbuffer[32];  //we'll store the SMS sender number in here


      if (! fona.getSMSSender(slot, callerIDbuffer, 31)) {
        Serial.println("Didn't find SMS message in slot!");
      }
      Serial.print(F("FROM: ")); Serial.println(callerIDbuffer);
```

Fig 4

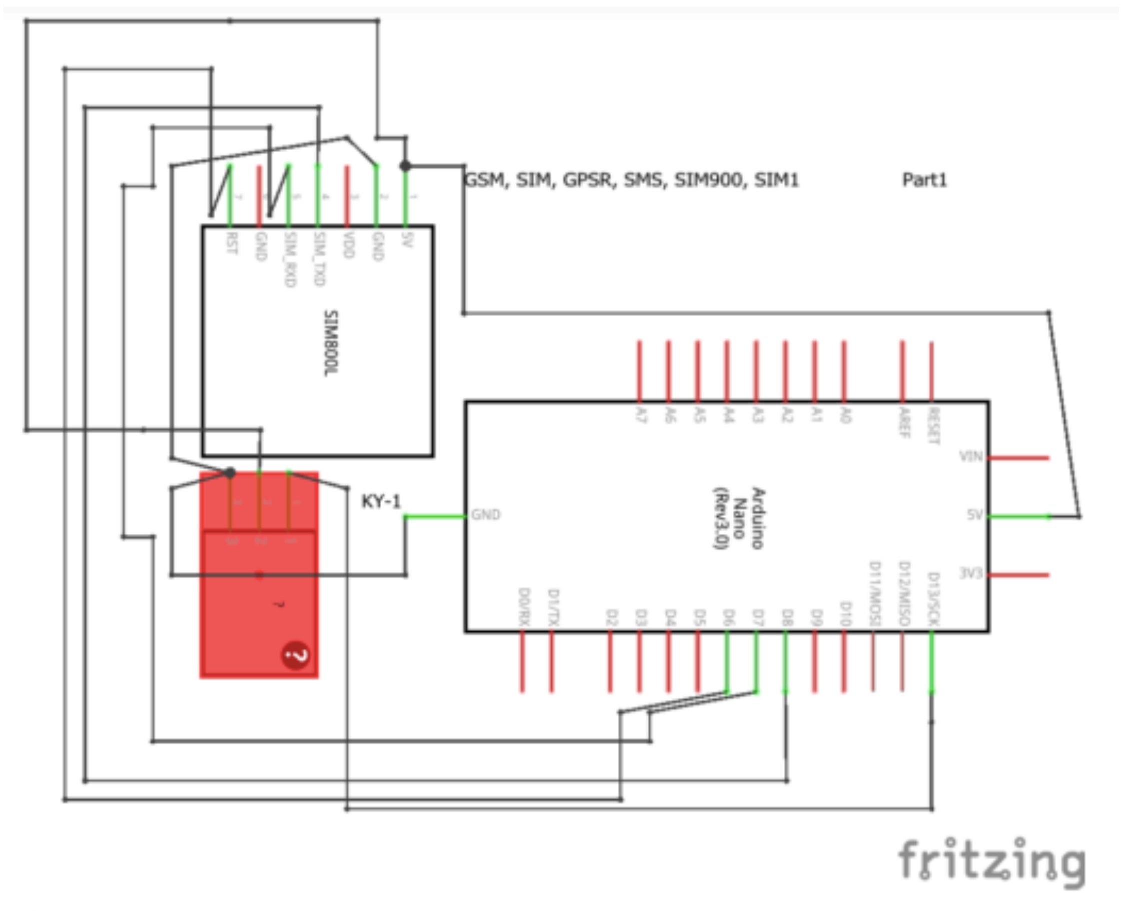
Now, we will upload the code to Arduino IDE and connect the components as given in the circuit diagram.

```
    if (! fona.getSMSSender(slot, callerIDbuffer, 31)) {
      Serial.println("Didn't find SMS message in slot!");
    }
    Serial.print(F("FROM: ")); Serial.println(callerIDbuffer);

      // Retrieve SMS value.
      uint16_t smslen;
      if (fona.readSMS(slot, smsBuffer, 250, &smslen)) { // pass in buffer and max len!
        Serial.println(smsBuffer);
        commands=smsBuffer;
        Serial.println(commands);
        if(commands=="lights on"){
          digitalWrite(13,1);
          }
        if(commands=="lights off"){
          digitalWrite(13,0);
          }
         if(commands=="fan on"){
          digitalWrite(12,1);
          }
        if(commands=="fan off"){
          digitalWrite(12,0);
          }

      }
```

Fig 5

Connection

Now connect the components as in pic below



Connection

Testing

Insert any SIM card (that supports 2G network) in the SIM800L module and connect any appliance through a relay. Now, send the message consisting of commands to the phone number (of the SIM card).

Here, I have used the lights on/off command for turning the lights on or off. You can set any command in the code, which you can control using SMS.

Download Source Code

27

RFID-BASED AUTOMATIC DESKTOP/MOBILE PHONE LOGIN DEVICE

BY ASHWINI KUMAR SINHA

We have so many login passwords to remember in order to access our laptops, emails or online banking. And forgetting password is something that we often face, and when we are in an urgent situation, it is quite annoying.

With cameras all around today, it is also not safe to type our password in public places. Now, you won't be required to remember any passwords to open your laptop/computer. The device we are going to build today will do the task for you.

This is a radio-frequency identification (RFID)-based automatic desktop login and password filling device. When we plugin this device in our laptop or mobile phone, it asks for the right RFID tag and when you put the unique RFID tag, it automatically fills the password and do the login or any tasks as programmed. When given wrong RFID tag, it simply rejects and ask for the right RFID tag. This means only the person who has the right RFID tag can access the system.

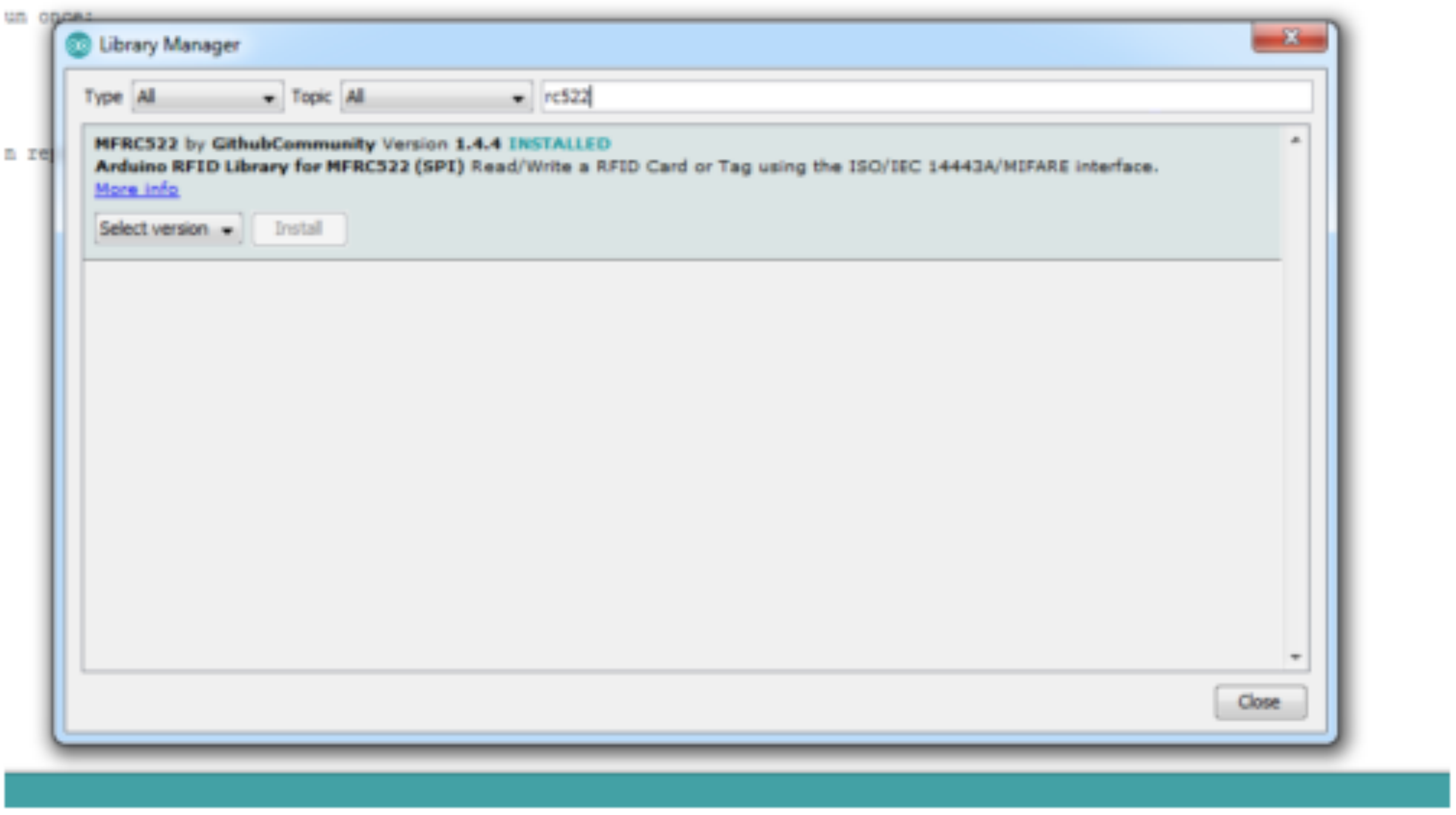
Note: - This is the first version of the project, we will soon update with more functions.

Components Required

- RFID RC522 Module kit
- Arduino Leonardo/Arduino Pro micro

Coding

Open the Arduino IDE and install the RC522 RFID library form library manager as in Fig 1.

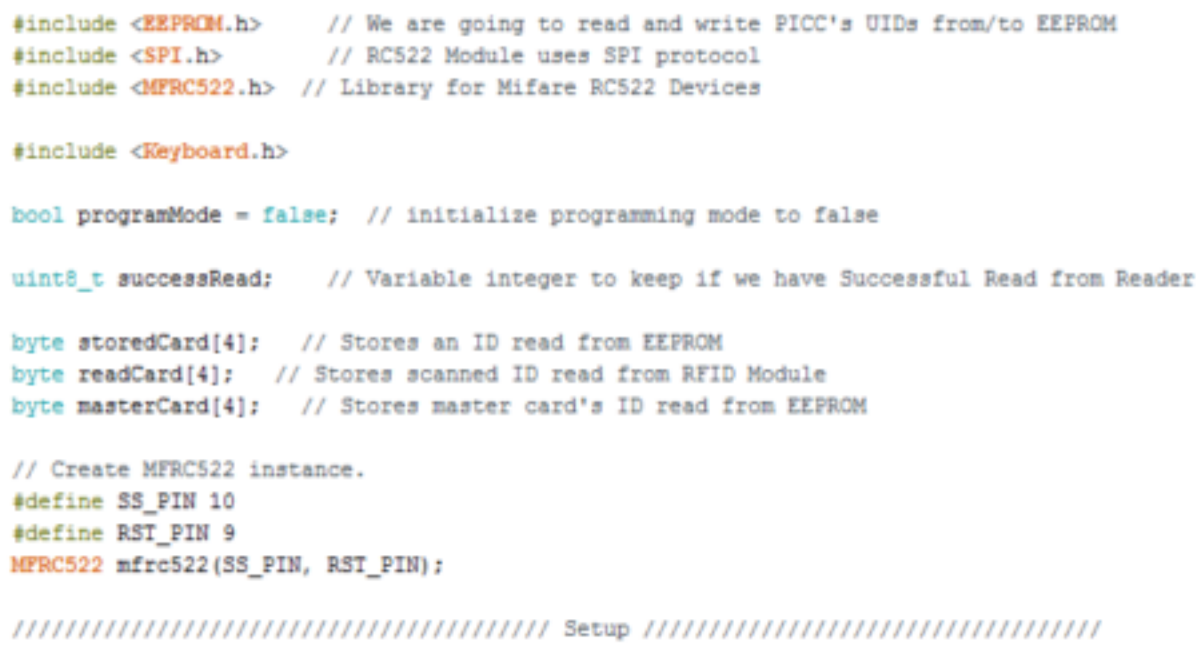


rc522

MFRC522 by GithubCommunity Version 1.4.4 INSTALLED
Arduino RFID Library for MFRC522 (SPI) Read/Write a RFID Card or Tag using the ISO/IEC 14443A/MIFARE interface.

Fig 1. Installing Library

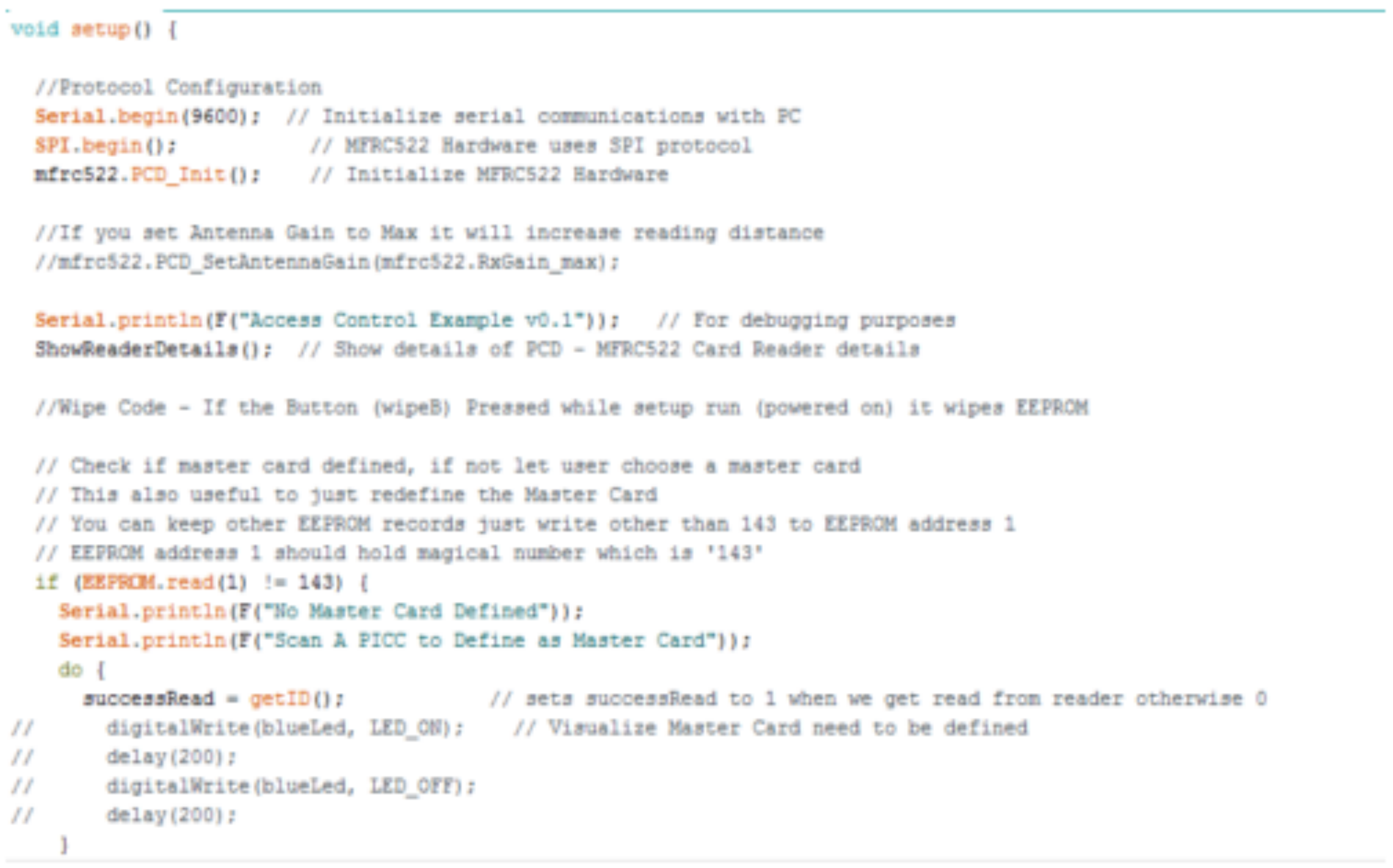
Now, let's understand the code. First, we have included the required library and then we create some variables to store the RFID card and tags UUID and values. Refer Fig 2.



```
#include <EEPROM.h>     // We are going to read and write PICC's UIDs from/to EEPROM
#include <SPI.h>        // RC522 Module uses SPI protocol
#include <MFRC522.h>  // Library for Mifare RC522 Devices

#include <Keyboard.h>

bool programMode = false;  // initialize programming mode to false

uint8_t successRead;    // Variable integer to keep if we have Successful Read from Reader

byte storedCard[4];   // Stores an ID read from EEPROM
byte readCard[4];   // Stores scanned ID read from RFID Module
byte masterCard[4];   // Stores master card's ID read from EEPROM

// Create MFRC522 instance.
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

///////////////////////////////////////// Setup ///////////////////////////////////
```

Fig 2. Arduino Code

Next we create a set-up function where the RFID and Spi are initialized. A statement then checks the card UUID to match it with the stored card. After that we make another statement that automatically types the password that is set in code i.e. Keyboard.println ("password").



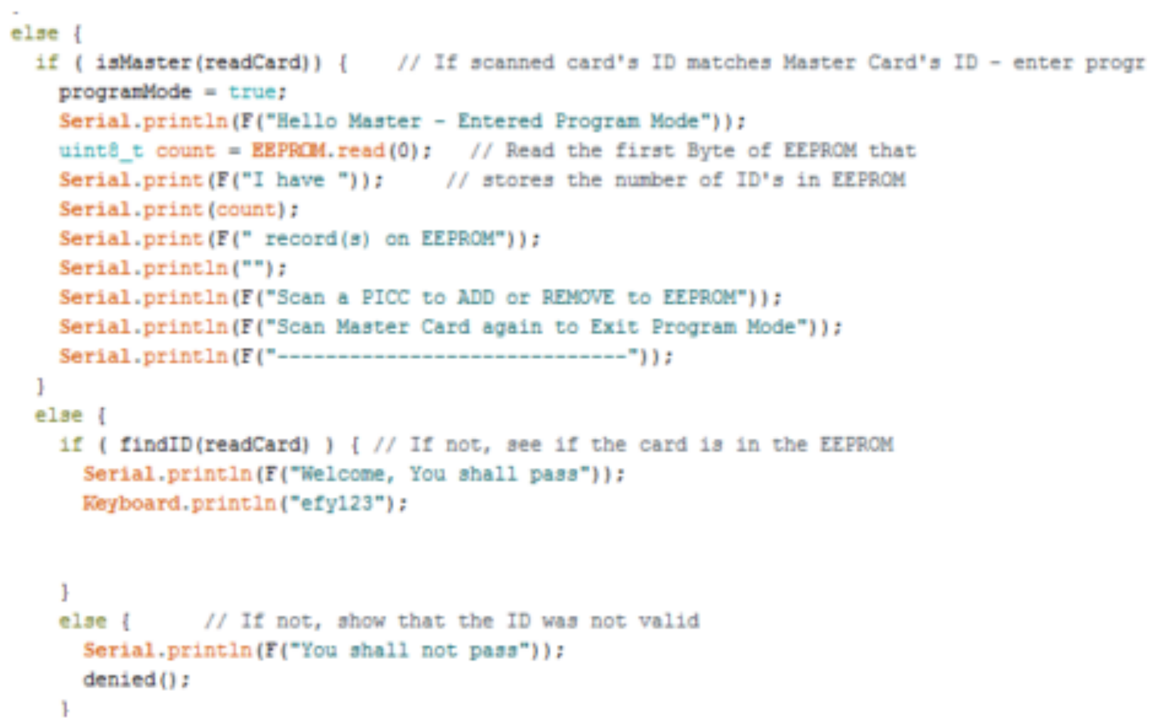
```
void setup() {

  //Protocol Configuration
  Serial.begin(9600);  // Initialize serial communications with PC
  SPI.begin();           // MFRC522 Hardware uses SPI protocol
  mfrc522.PCD_Init();    // Initialize MFRC522 Hardware

  //If you set Antenna Gain to Max it will increase reading distance
  //mfrc522.PCD_SetAntennaGain(mfrc522.RxGain_max);

  Serial.println(F("Access Control Example v0.1"));   // For debugging purposes
  ShowReaderDetails();  // Show details of PCD - MFRC522 Card Reader details

  //Wipe Code - If the Button (wipeB) Pressed while setup run (powered on) it wipes EEPROM

  // Check if master card defined, if not let user choose a master card
  // This also useful to just redefine the Master Card
  // You can keep other EEPROM records just write other than 143 to EEPROM address 1
  // EEPROM address 1 should hold magical number which is '143'
  if (EEPROM.read(1) != 143) {
    Serial.println(F("No Master Card Defined"));
    Serial.println(F("Scan A PICC to Define as Master Card"));
    do {
      successRead = getID();             // sets successRead to 1 when we get read from reader otherwise 0
//      digitalWrite(blueLed, LED_ON);    // Visualize Master Card need to be defined
//      delay(200);
//      digitalWrite(blueLed, LED_OFF);
//      delay(200);
    }
```

Fig 3. Arduino code initializing RFID

Download the code and make changes as stated and then upload the code to Arduino Leonardo.

Connection

After uploading the code, make the connection as illustrated below. (Refer Fig 5.)



```
else {
  if ( isMaster(readCard)) {    // If scanned card's ID matches Master Card's ID - enter progr
    programMode = true;
    Serial.println(F("Hello Master - Entered Program Mode"));
    uint8_t count = EEPROM.read(0);   // Read the first Byte of EEPROM that
    Serial.print(F("I have "));     // stores the number of ID's in EEPROM
    Serial.print(count);
    Serial.print(F(" record(s) on EEPROM"));
    Serial.println("");
    Serial.println(F("Scan a PICC to ADD or REMOVE to EEPROM"));
    Serial.println(F("Scan Master Card again to Exit Program Mode"));
    Serial.println(F("-----------------------------"));
  }
  else {
    if ( findID(readCard) ) { // If not, see if the card is in the EEPROM
      Serial.println(F("Welcome, You shall pass"));
      Keyboard.println("efy123");


    }
    else {      // If not, show that the ID was not valid
      Serial.println(F("You shall not pass"));
      denied();
    }
```

Fig 4. Arduino code checking the RFID tag and typing password

Arduino Pro Micro	**RS522 RFID**
RST	Reset
Pin 10	SDA
Pin 16	MOSI
Pin 14	MISO
Pin 15	SCK
VCC	VCC
GND	GND

Testing

Now, plug the Leonardo to your PC and then open the serial monitor. After that bring the master card provided with RFID Kit near the RFID reader module. When it reads the master card, the program turns into card storage mode. As we bring the tag or any NFC card near RFID reader, the program automatically reads the card and store its UUID in EEPROM.

Whenever you want to login to your PC, just plug the device in and bring your unique RFID tag near it. The device will automatically fill the password and do the login for you.

This is like a physical key for digital world. You can also delete/change the tag. To do so, bring the master card near the RFID reader to enter in programming mode. Then bring the tag to add or remove for verification.

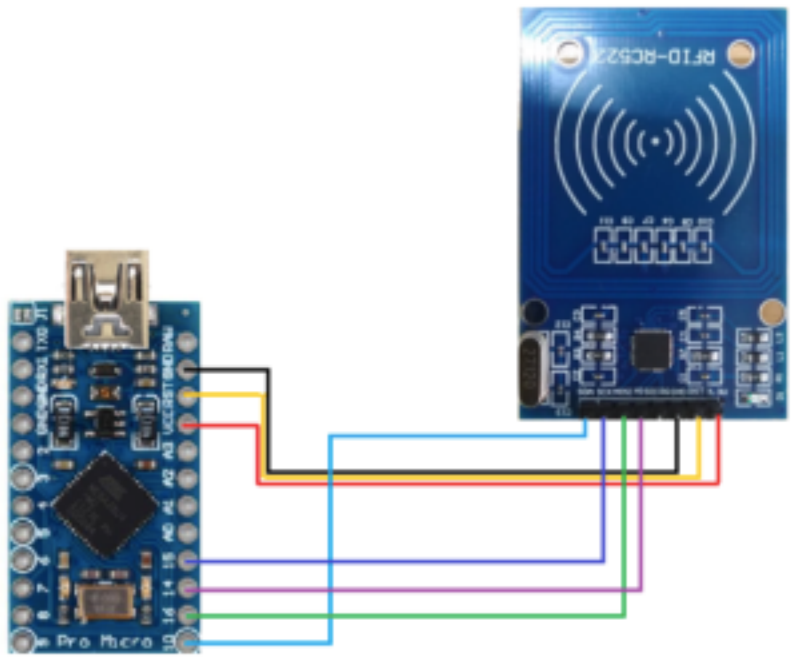


Fig 5. Connection

Download Source Code

Download Source Code With Oled

You can watch Video for this DIY here: https://youtu.be/ServyW75oCk

28

FACE RECOGNITION USING RASPBERRY PI

BY BISWAJIT DAS

To secure our assets and protect our privacy, there is a strong demand for user-friendly biometric security systems. There are various types of biometric systems that use signatures, fingerprints, voice, hand geometry, ear geometry, face detection and so on. Among these, face recognition appears to be quite exciting and is catching attention.

Circuit and working

This project requires Raspberry Pi B+, Raspberry Pi camera, a pushbutton switch, a relay and some miscellaneous components.

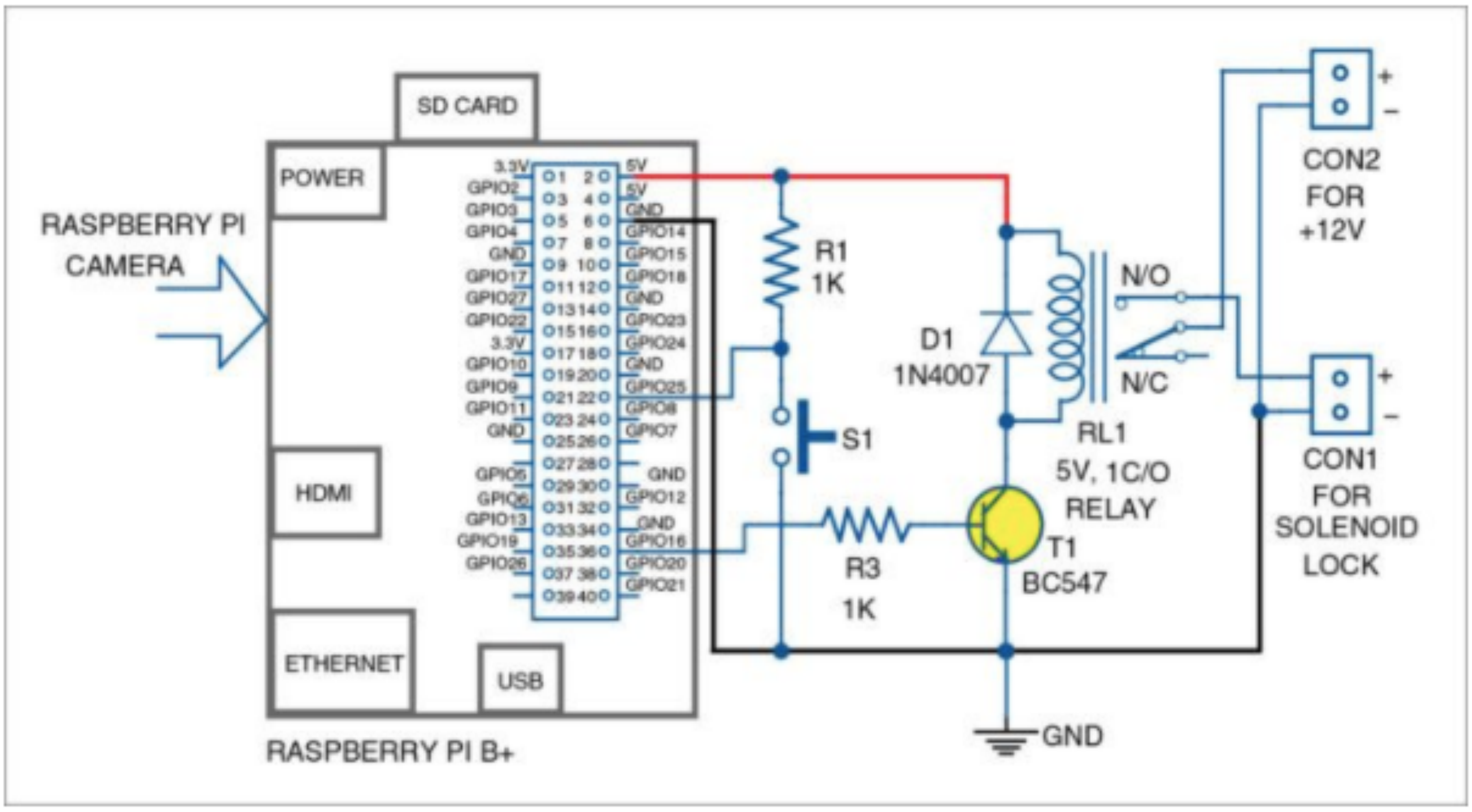


Fig. 1: Circuit diagram of the face-recognition system using Raspberry Pi

Face images are captured through Raspberry Pi camera and stored in a database in Raspberry Pi. To capture your face image, place yourself in front of the Pi camera and press pushbutton switch S1. The image of your face will get stored in the database. Next time when you face Pi camera and press S1, your face will be recognised, relay RL1 will be energised and your electrical load/solenoid will be activated.

Software

This project uses OpenCV computer vision library to perform face detection and recognition.

First, install OpenCV dependencies. Compiling OpenCV on Raspberry Pi may take about five hours (depending on your system and network speed). So make sure you have sufficient time to start the process before proceeding.

Power on Raspberry Pi, open the terminal, set up Wi-Fi and execute the following commands:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install build-essential
cmake pkg-config python-dev libgtk2.0-
dev libgtk2.0 zlib1g-dev libpng-dev
libjpeg-dev libtiff-dev libjasper-dev
libavcodec-dev swig unzip
```

Select yes for all options and wait for the libraries and dependencies to be installed.

Now, unzip OpenCV directory by executing the following commands:

```
$ wget
http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.4.9/o
pencv-2.4.9.zip
$ unzip opencv-2.4.9.zip
```

Change the directory and execute cmake command as given below to build the makefile:

```
$ cd opencv-2.4.9
$ cmake -DCMAKE_BUILD_TYPE=RELEASE
-DCMAKE_INSTALL_PREFIX=/usr/local
-DBUILD_PERF_TESTS=OFF -DBUILD_opencv_
gpu=OFF -DBUILD_opencv_ocl=OFF
```

Compile the project by executing the command given below:

```
$ make
```

It may take about five hours for compilation.

Install the compiled OpenCV libraries by executing the following command:

```
$ sudo make install
```

The latest version of OpenCV is now installed on your Raspberry Pi.

Face-recognition code is written in Python, so some dependencies have to be installed using the following commands:

```
$ sudo apt-get install python-pip
$ sudo apt-get install python-dev
$ sudo pip install picamera
$ sudo pip install rpio
```

After OpenCV and Python dependencies are installed, the project can be tested in three major steps as explained below.

Software testing

This project uses EIGENFACES ALGORITHM IN OPENCV to perform face recognition. To use this algorithm, create a set of training data with pictures of faces that are allowed to trigger the relay.

Follow the steps given below:

Execute the following command to run capture-positives script to find a single face image:

```
$ sudo python capture-positives.py
```

Wait for some time and observe the terminal until you see Press Button instruction. Press S1 to capture your face image. If the script detects a single face, it will crop and save the training image in positive sub-directory.

If the script cannot detect a face, or detects multiple faces, error message 'Could not detect single face! Check the image in capture.pgm' will be displayed. It is recommended to maintain a distance of about 0.5 metres from the camera while taking a picture.

Press Ctrl+C to stop the script. Open capture.pgm file to view the last captured image.

Check the face in the database and train the face recogniser by running train.py code:

```
$ sudo python train.py
```

Training the face-recognition model on Raspberry Pi will take about ten minutes. Once training is complete, you will see mean.png and positive_eigenface.png files to visualise Eigenfaces of the model.

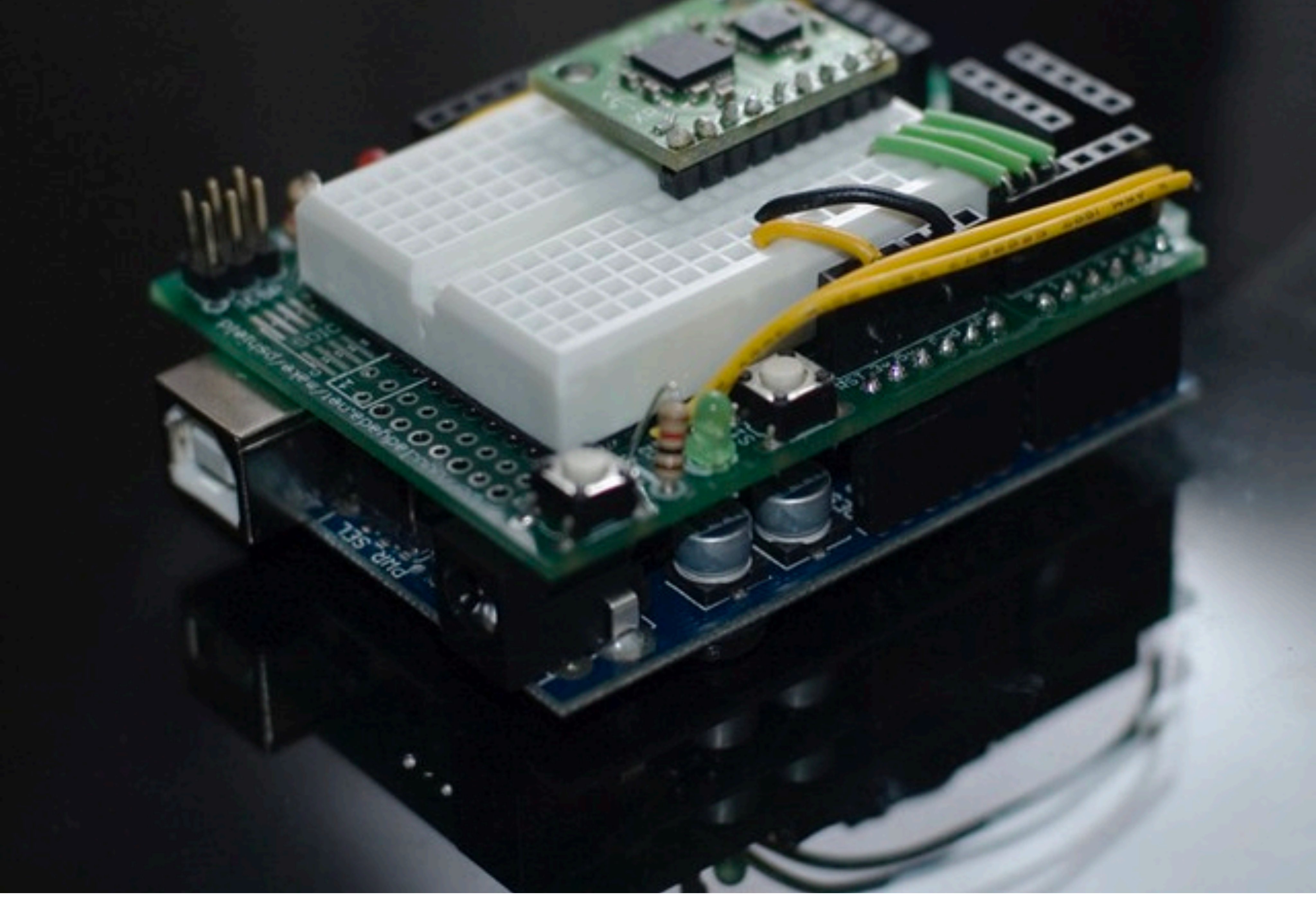
Now, test the face recogniser to recognise the face trained earlier. Execute the following command:

```
$ sudo python box.py
```

Observe the terminal to see Press Button instruction. Aim the camera at your face and press S1. You should see a message ‘Button pressed, looking for face...’ on the terminal. After a few seconds, if the face is recognised, you will see the message ‘Recognised face!’ and relay RL1 will be energised.

You can connect an electrical load across the relay contacts at CON1. A 12V solenoid lock may be used as electrical load during testing. If your face is recognised or detected, relay is energised, solenoid lock is activated and the secured door gets opened.

Download source code



29

SMART INCLINOMETER & MULTI-PURPOSE MEASUREMENT DEVICE

BY ASHWINI KUMAR SINHA

Today, we are going to make an ‘all-in-one’ measurement device/instrument, which consists of a digital inclinometer, compass and protractor. It also includes a health monitoring system that can provide data about temperature, oxygen levels and heartbeat.

All these features come in a very small form factor for ease of use and portability.

For inclinometer and smart protractor, we are going to use a tiny mpu6050 sensor. It can also be used to collect temperature-related information. For health monitoring, a MAX30100 sensor is used. It's red and IR lights along with photosensors can determine heartbeat and oxygen levels.

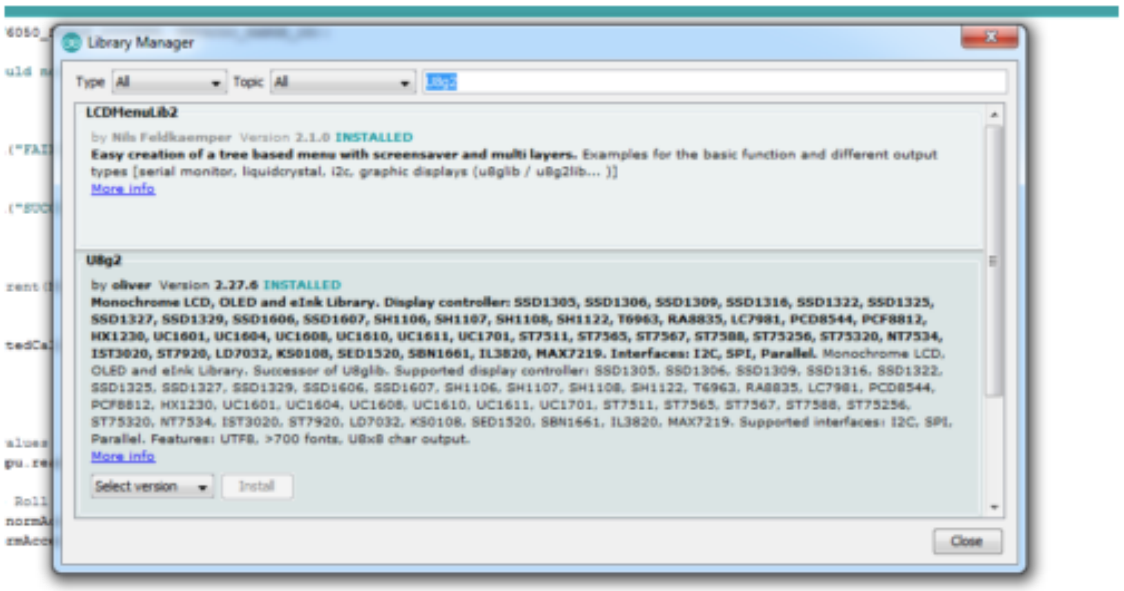
So, let's start our project by collecting the following components.

Bill Of Materials

Component Name	Quantity	Description	Cost Approx. In INR
Arduino Pro Micro	1	For Programming	300
MAX30100	1	Heart Beat Sensing	300
MPU6050	1	Acceleration measurement	200
OLED SSD1306	1	Display	400
Battery	1	3.3V - 5V	200
Wires	30cm	For Connection	30
Total Cost			**1430**

Pre-requisites

Assuming that you already have the Arduino IDE installed in your computer and connected to the internet as well, go to the sketch menu and click on the library manager. A window having a list of different libraries shows up. Now search the following libraries and install them one by one. (Refer fig 2).



LCDMenuLib2
by Nils Feldkaemper Version 2.1.0 INSTALLED
Easy creation of a tree based menu with screensaver and multi layers. Examples for the basic function and different output types [serial monitor, liquidcrystal, i2c, graphic displays (u8glib / u8g2lib...)]
More info

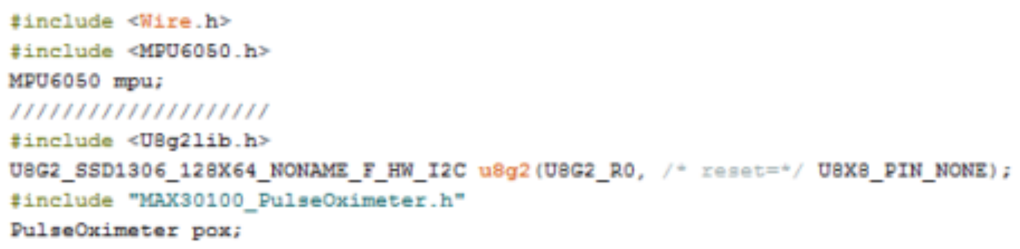
U8g2
by oliver Version 2.27.6 INSTALLED
Monochrome LCD, OLED and eInk Library. Display controller: SSD1305, SSD1306, SSD1309, SSD1316, SSD1322, SSD1325, SSD1327, SSD1329, SSD1606, SSD1607, SH1106, SH1107, SH1108, SH1122, T6963, RA8835, LC7981, PCD8544, PCF8812, HX1230, UC1601, UC1604, UC1608, UC1610, UC1611, UC1701, ST7511, ST7565, ST7567, ST7588, ST75256, ST75320, NT7534, IST3020, ST7920, LD7032, KS0108, SED1520, SBN1661, IL3820, MAX7219. Interfaces: I2C, SPI, Parallel. Monochrome LCD, OLED and eInk Library. Successor of U8glib. Supported display controller: SSD1305, SSD1306, SSD1309, SSD1316, SSD1322, SSD1325, SSD1327, SSD1329, SSD1606, SSD1607, SH1106, SH1107, SH1108, SH1122, T6963, RA8835, LC7981, PCD8544, PCF8812, HX1230, UC1601, UC1604, UC1608, UC1610, UC1611, UC1701, ST7511, ST7565, ST7567, ST7588, ST75256, ST75320, NT7534, IST3020, ST7920, LD7032, KS0108, SED1520, SBN1661, IL3820, MAX7219. Supported interfaces: I2C, SPI, Parallel. Features: UTF8, >700 fonts, U8x8 char output.
More info

Fig 2.

- Arduino MPU6050
- U8g2
- MAX30100

Coding

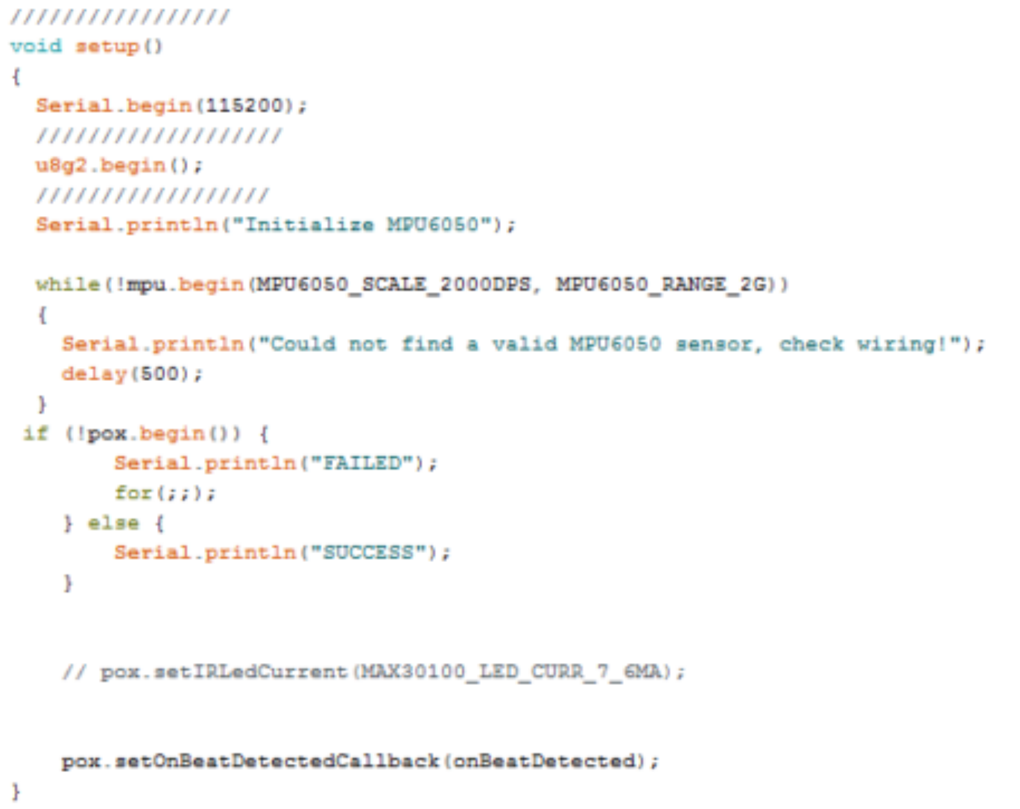
Firstly, we will include the required libraries in the code. (Refer fig 3)



```
#include <Wire.h>
#include <MPU6050.h>
MPU6050 mpu;
/////////////////////
#include <U8g2lib.h>
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);
#include "MAX30100_PulseOximeter.h"
PulseOximeter pox;
```

Fig 3.

Secondly, we will create a setup function to initialize the I2C-based sensors and OLED display .(Refer fig 4)



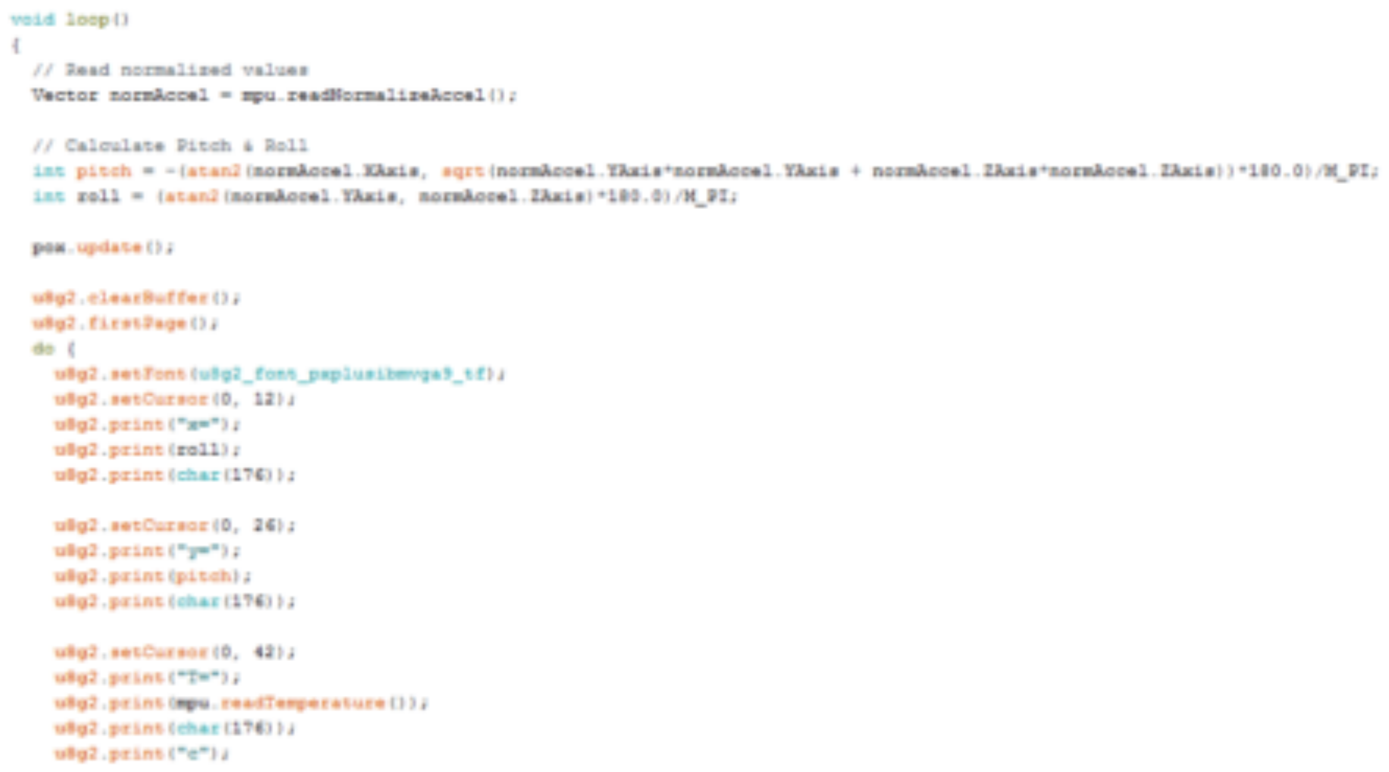
```
//////////////////
void setup()
{
  Serial.begin(115200);
  ///////////////////
  u8g2.begin();
  //////////////////
  Serial.println("Initialize MPU6050");

  while(!mpu.begin(MPU6050_SCALE_2000DPS, MPU6050_RANGE_2G))
  {
    Serial.println("Could not find a valid MPU6050 sensor, check wiring!");
    delay(500);
  }
 if (!pox.begin()) {
        Serial.println("FAILED");
        for(;;);
    } else {
        Serial.println("SUCCESS");
    }


    // pox.setIRLedCurrent(MAX30100_LED_CURR_7_6MA);


    pox.setOnBeatDetectedCallback(onBeatDetected);
}
```

Fig 4.

Thirdly, we will create a loop function that will convert x,y,z axes readings of the accelerometer into angle of rotation. By writing a set of code, this result can be displayed on an OLED screen. The readings from the MAX30100 sensor can also be displayed on the OLED screen. (Refer fig 5)

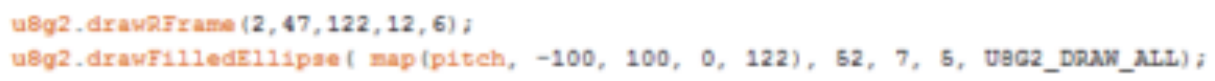
Lastly, we will write a code that will create a rectangular frame and a small circle within that. The y-axis of the circle is mapped with the angle at y-axis of mpu6050 reading. This will allow the circle to move according to the angle of inclination. (Refer fig 6)

Now our code is ready, which you can download and upload to Arduino.



```
void loop()
{
  // Read normalized values
  Vector normAccel = mpu.readNormalizeAccel();

  // Calculate Pitch & Roll
  int pitch = -(atan2(normAccel.XAxis, sqrt(normAccel.YAxis*normAccel.YAxis + normAccel.ZAxis*normAccel.ZAxis))*180.0)/M_PI;
  int roll = (atan2(normAccel.YAxis, normAccel.ZAxis)*180.0)/M_PI;

  pox.update();

  u8g2.clearBuffer();
  u8g2.firstPage();
  do {
    u8g2.setFont(u8g2_font_pxplusibmvga9_tf);
    u8g2.setCursor(0, 12);
    u8g2.print("x=");
    u8g2.print(roll);
    u8g2.print(char(176));

    u8g2.setCursor(0, 26);
    u8g2.print("y=");
    u8g2.print(pitch);
    u8g2.print(char(176));

    u8g2.setCursor(0, 42);
    u8g2.print("T=");
    u8g2.print(mpu.readTemperature());
    u8g2.print(char(176));
    u8g2.print("c");
```

Fig 5.



```
u8g2.drawRFrame(2,47,122,12,6);
u8g2.drawFilledEllipse( map(pitch, -100, 100, 0, 122), 52, 7, 5, U8G2_DRAW_ALL);
```

Fig 6.

Connections

After uploading the code, make the following connections. (Refer fig 7)

Arduino Pro Micro	**MPU6050,Max30100,SSD1306**
SCL(Pin 2)	SCL
SDA(Pin3)	SDA
VCC	VCC
GND	GND

Connect the +ve terminal of the 3.3V - 5V battery to the VCC pin and -ve to the GND pin.

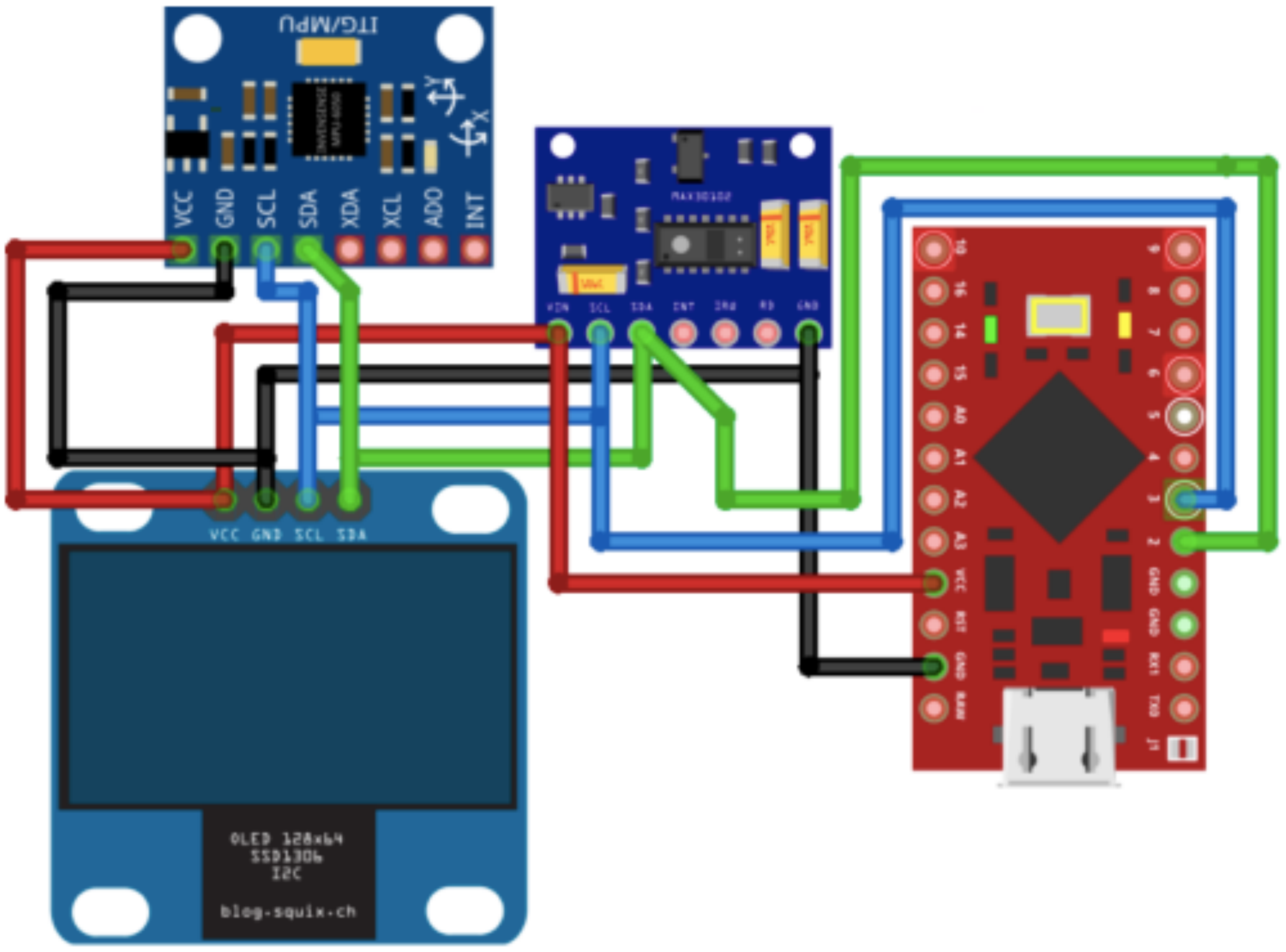


Fig 7. Circuit diagram made in Fritzing

Testing

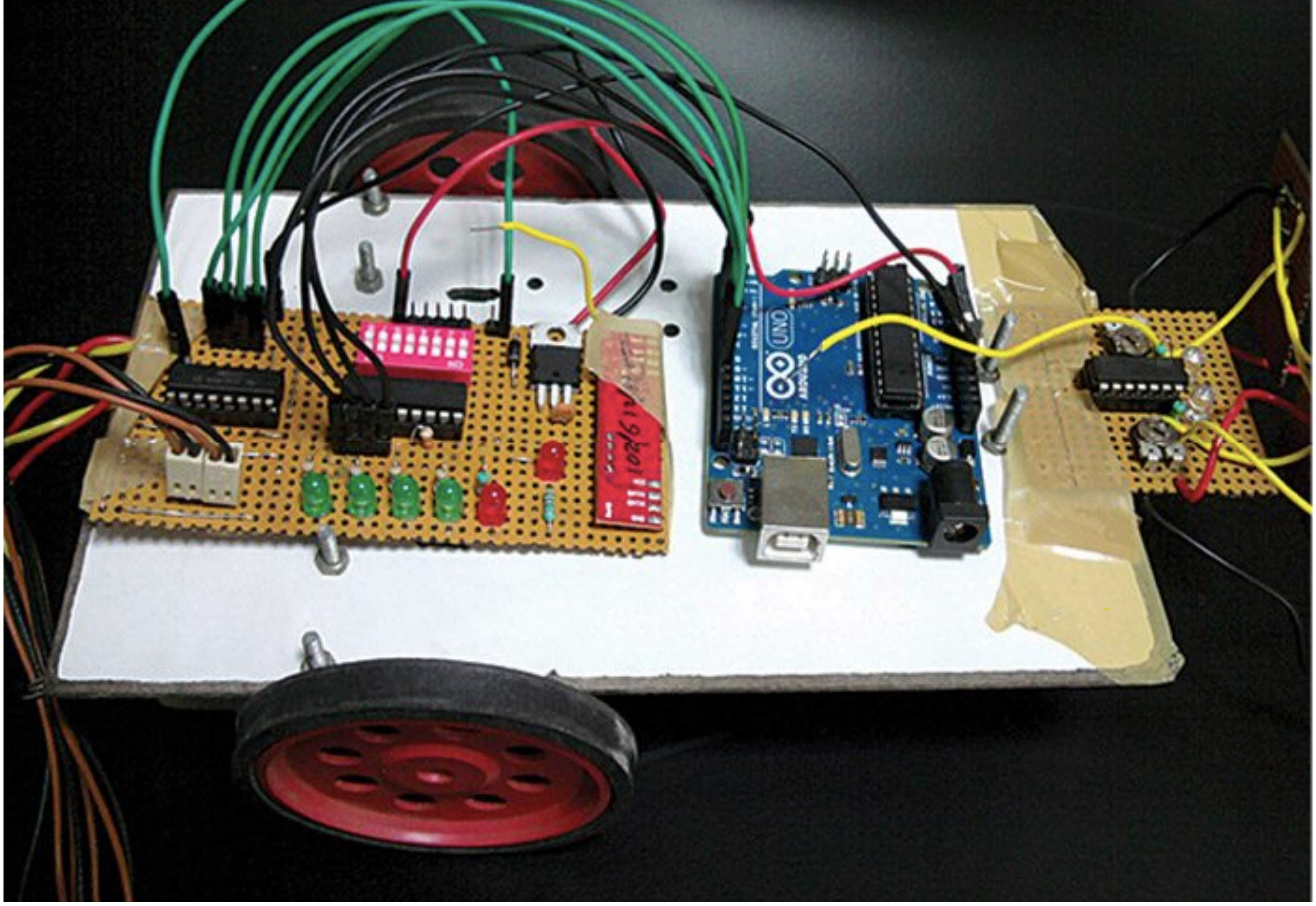
To measure the inclination of any surface or to determine whether it is perfectly flat or not, place the measuring device on that surface. If x = 0 and y=0 readings are obtained, then the surface is flat.

If the surface is inclined, the circle moves accordingly and displays the angle of inclination. At the same time, the device displays the surrounding temperature.

In order to get a reading of your health, put your finger on the MAX sensor and wait a few seconds. After calculating the readings from the sensor, the device will now show your heartbeat in BPS and oxygen level in percentage.

Download Source Code: click here

You can watch Video for this DIY here: https://youtu.be/ZKg1jObK2Ws



30

RF CONTROLLED ROBOT

BY ROBIN CHALANA

Here we present a simple Arduino based RF controlled robot that can be driven remotely. This robot can be built very quickly on a small budget. The RF remote control provides the advantage of a good controlling range (up to 100 metres with proper antennae) besides being omnidirectional.

Arduino based RF controlled robot circuit

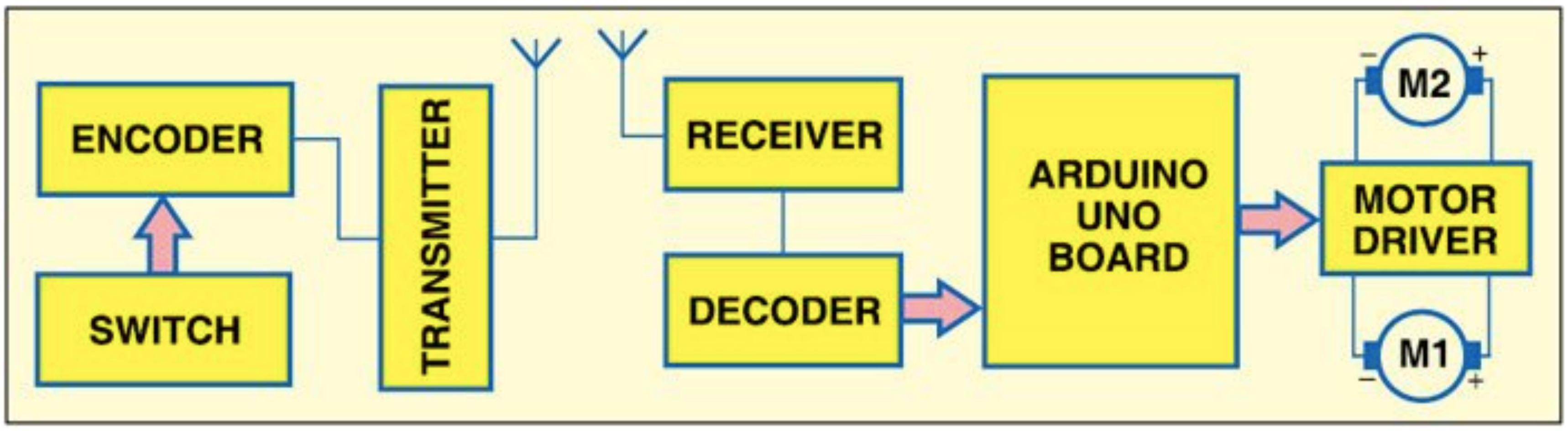


Fig. 1: Block diagram of Arduino based RF controlled robot

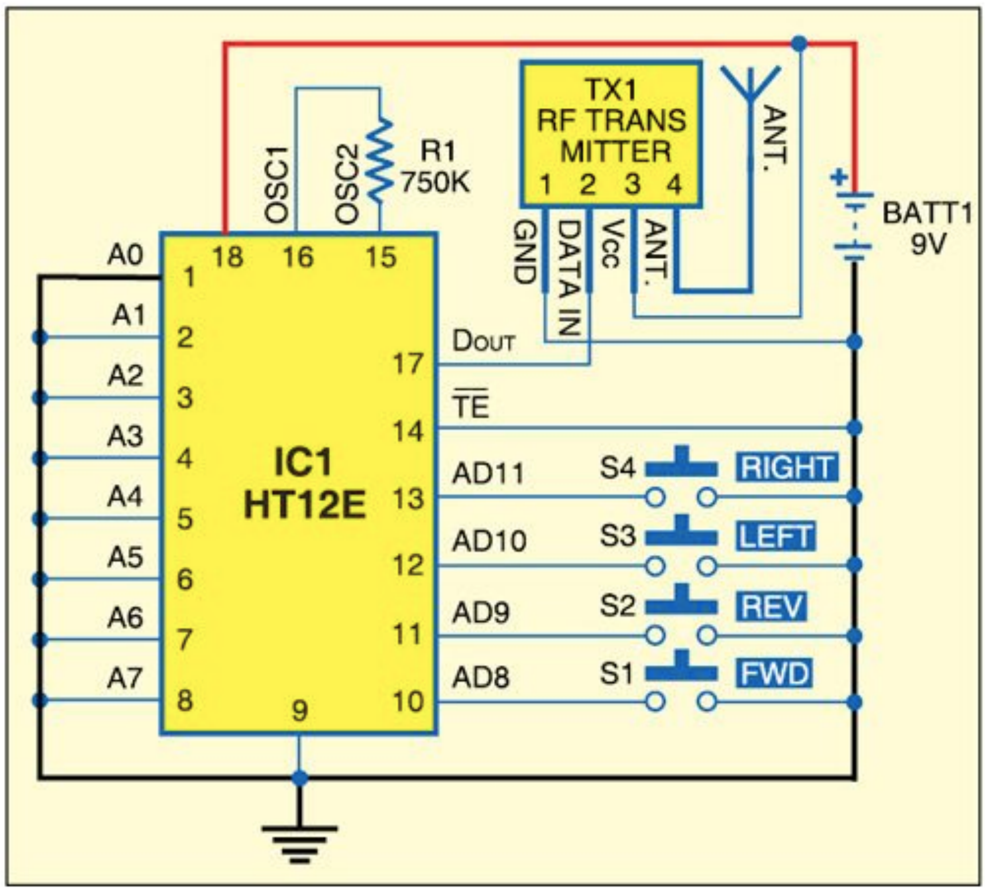


Fig. 2: Circuit of transmitter section

The block diagram of the robot is shown in Fig. 1. It has two major sections: (a)transmitter and (b)receiver and motor driver. The transmitter circuit (Fig. 2) is built around encoder IC HT12E (IC1), 433MHz RF transmitter module (TX1) and a few discrete components. The receiver and motor driver circuit (Fig. 3) is built around Arduino UNO board (BOARD1), decoder IC HT12D (IC2), 433MHz RF receiver module (RX1), motor driver IC L293D (IC3), regulator IC 7805 (IC4) and a few discrete components.

Arduino UNO board

The heart of the robot is Arduino UNO board. Arduino is an Open Source electronics prototyping platform based on flexible, easy-to-use hardware and software. It is intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

The Arduino Uno board is based on the ATmega328 microcontroller. It consists of 14 digital input/output pins, six analogue inputs, a USB connection for programming the on-board microcontroller, a power jack, an ICSP header and a reset button. It is operated with a 16MHz crystal oscillator. It contains everything needed to support the microcontroller. It is very user-friendly; simply connect it to a computer with a USB cable to get started. The microcontroller on the board is programmed using the Arduino programming language and the Arduino development environment.

Technical Specifications of Arduino UNO Board

Details	Specifications
Microcontroller	ATmega328
Operating voltage	5V
Input voltage—recommended	7-12V
Input voltage—limits	6-20V
Digital I/O pins	14 (of which six provide PWM output)
Analogue input pins	6
DC current per I/O pin	40 mA
DC current for 3.3V pin	50 mA
Flash memory	32 kB (ATmega328), of which 0.5 kB is used by bootloader
SRAM	2 kB (ATmega328)
EEPROM	1 kB (ATmega328)
Clock speed	16 MHz

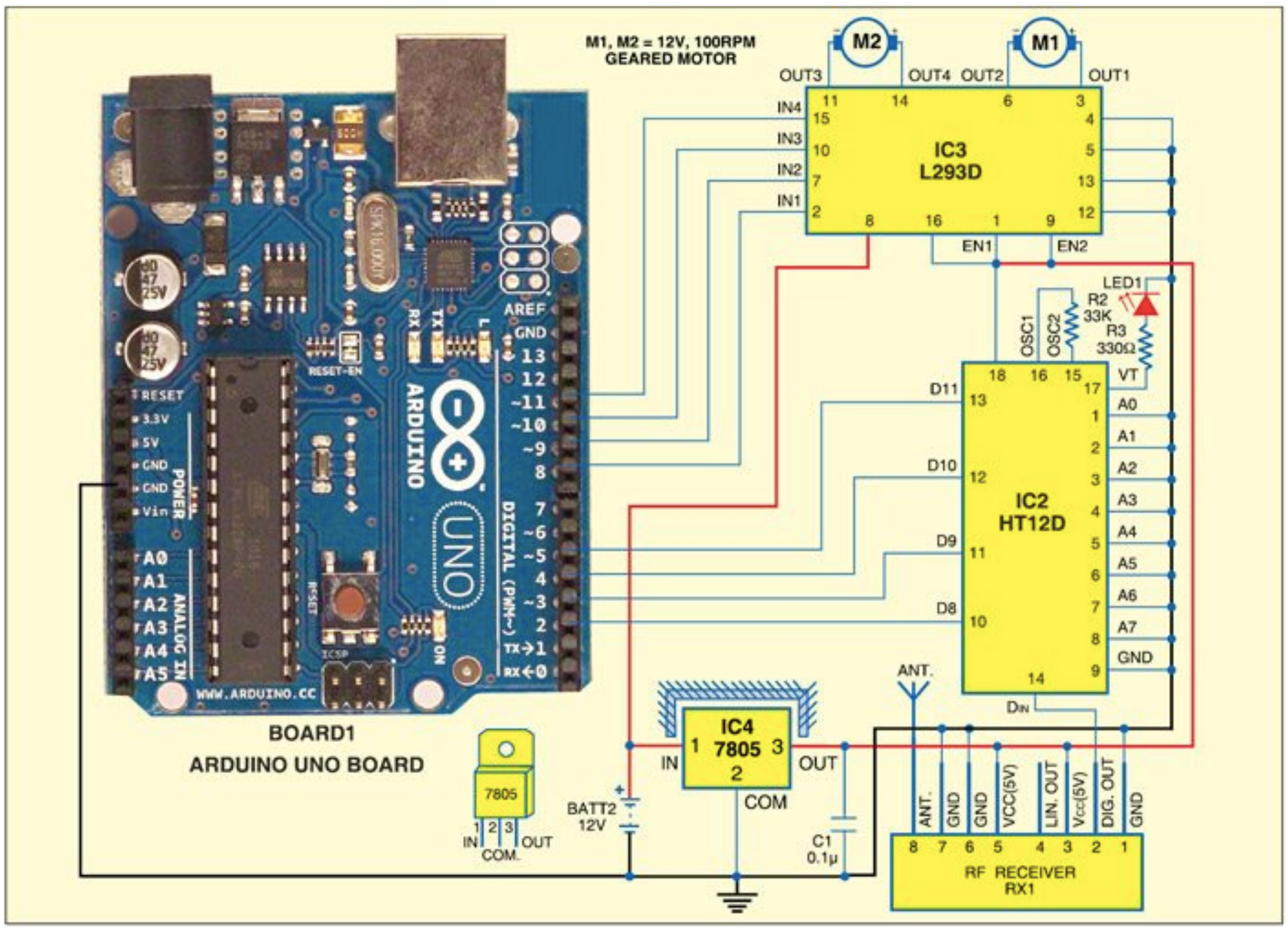
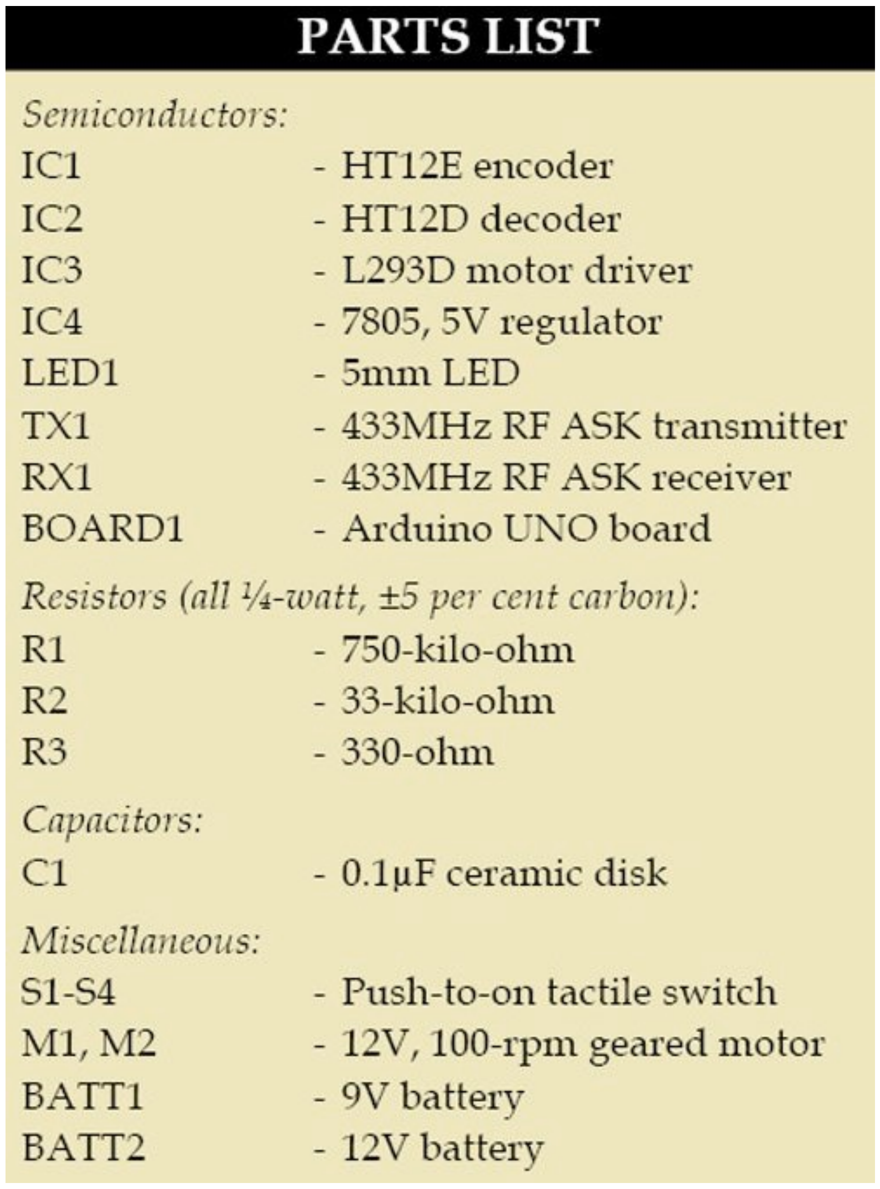


Fig. 3: Circuit of receiver and motor driver

Remote control

For controlling the robot remotely, Holteks' encoder-decoder pair (HT12E and HT12D) together with a 433MHz transmitter-receiver pair is used.



PARTS LIST

Semiconductors:	
IC1	- HT12E encoder
IC2	- HT12D decoder
IC3	- L293D motor driver
IC4	- 7805, 5V regulator
LED1	- 5mm LED
TX1	- 433MHz RF ASK transmitter
RX1	- 433MHz RF ASK receiver
BOARD1	- Arduino UNO board
Resistors (all ¼-watt, ±5 per cent carbon):	
R1	- 750-kilo-ohm
R2	- 33-kilo-ohm
R3	- 330-ohm
Capacitors:	
C1	- 0.1µF ceramic disk
Miscellaneous:	
S1-S4	- Push-to-on tactile switch
M1, M2	- 12V, 100-rpm geared motor
BATT1	- 9V battery
BATT2	- 12V battery

HT12E and HT12D are CMOS ICs with working voltage ranging from 2.4V to 12V. Encoder HT12E has eight address and another four address/data lines. The data set on these twelve lines (address and address/data lines) is serially transmitted when transmit-enable pin TE is taken low. The data output appears serially on DOUT pin.

The data is transmitted four times in succession. It consists of differing lengths of positive-going pulses for '1' and '0,' the pulse-width for '0' being twice the pulse-width for '1.' The frequency of these pulses may lie between 1.5 and 7 kHz depending on the resistor value between OSC1 and OSC2 pins.

The internal oscillation frequency of decoder HT12D is 50 times the oscillation frequency of encoder HT12E. The HT12D receives the data from the HT12E on its DIN pin serially. If the address part of the data received matches the levels on A0 through A7 pins four times in succession, the valid transmission (VT) pin is taken high. The data on pins AD8 through AD11 of the HT12E appears on pins D8 through D11 of the HT12D. Thus the device acts as a receiver of 4-bit data (16 possible codes) with 8-bit addressing (256 possible channels).

Transmitter circuit

Switches S1, S2, S3 and S4 are interfaced with AD8 through AD11 of encoder HT12E for forward (FWD), reverse (REV), left (LEFT) and right (RIGHT) motions, respectively. Resistor R1 is connected between oscillator pins 15 and 16 to set the transmitter frequency.

HT12E is permanently enabled for transmission by connecting its TE pin to ground. When any switch, say, S1, is pressed, the corresponding data is serially transmitted from DOUT pin through the RF ASK transmitter module. A 9V battery is used to power the circuit.

Receiver and motor driver circuit. Assuming that address pins on the encoder and the decoder are identical, when any of the switches on the transmitter (marked as FWD, REV, RIGHT, LEFT) is pressed, the corresponding data pin of the decoder goes low. The data outputs from D8 through D11 of HT12D (IC2) are fed to pins 2 through 5 of Arduino UNO board to generate appropriate logic outputs from pins 8 through 11 of Arduino UNO board.

Outputs from pins 8 through 11 of Arduino Uno board are fed to IN1 through IN4 of L293D (IC3) to drive both the motors M1 and M2 as shown in Fig. 3. Outputs OUT1 and OUT2 drive motor M1, and outputs OUT3 and OUT4 drive motor M2. Enable pins EN1 (pin 1) and EN2 (pin 9) are connected to Vcc for always enabled output. Regulator 7805 (IC4) provides regulated 5V to the receiver.

Construction

A single side PCB for the RF transmitter (Fig. 2) is shown in Fig. 4 and its component layout in Fig. 5. The PCB for the receiver (Fig. 3) is shown in Fig. 6 and its component layout in Fig. 7. A suitable connector arrangement has been made on the RF receiver PCB in order to extend connections to the drive motors and the battery mounted on the chassis of the RF robot.

Download source code: click here

Software

The source code file (RFROBOT.INO) for this project is listed at the end of this article. The Arduino Uno is programmed with Arduino IDE software. The ATmega328 on Arduino Uno comes pre-burned with a bootloader that allows you to upload new code to it without using an external hardware programmer. It communicates using the original STK500 protocol. You can also bypass the bootloader and program the microcontroller through the ICSP (in-circuit serial programming) header but using the bootloader makes the programming quick and easy.

Select Arduino Uno from the Tools→Board menu (according to the microcontroller on your board) in the Arduino IDE and burn the program through the standard USB port in the computer.

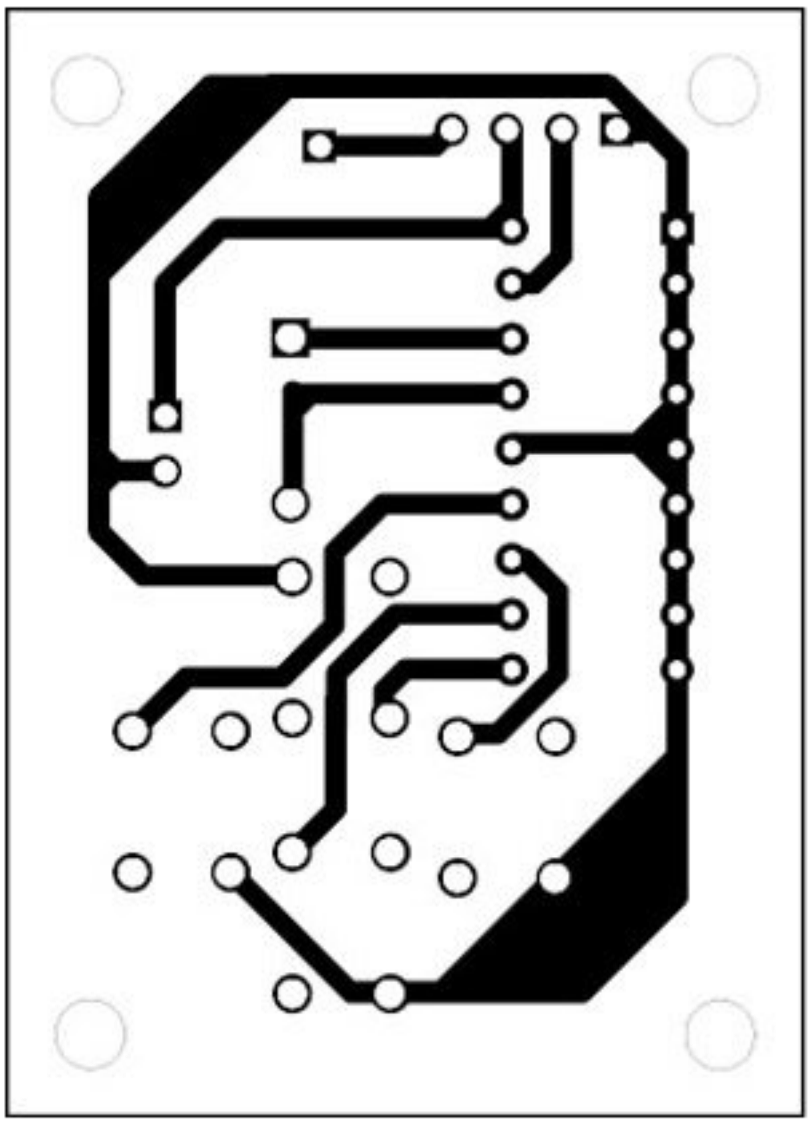


Fig. 4: A single side PCB for the RF transmitter

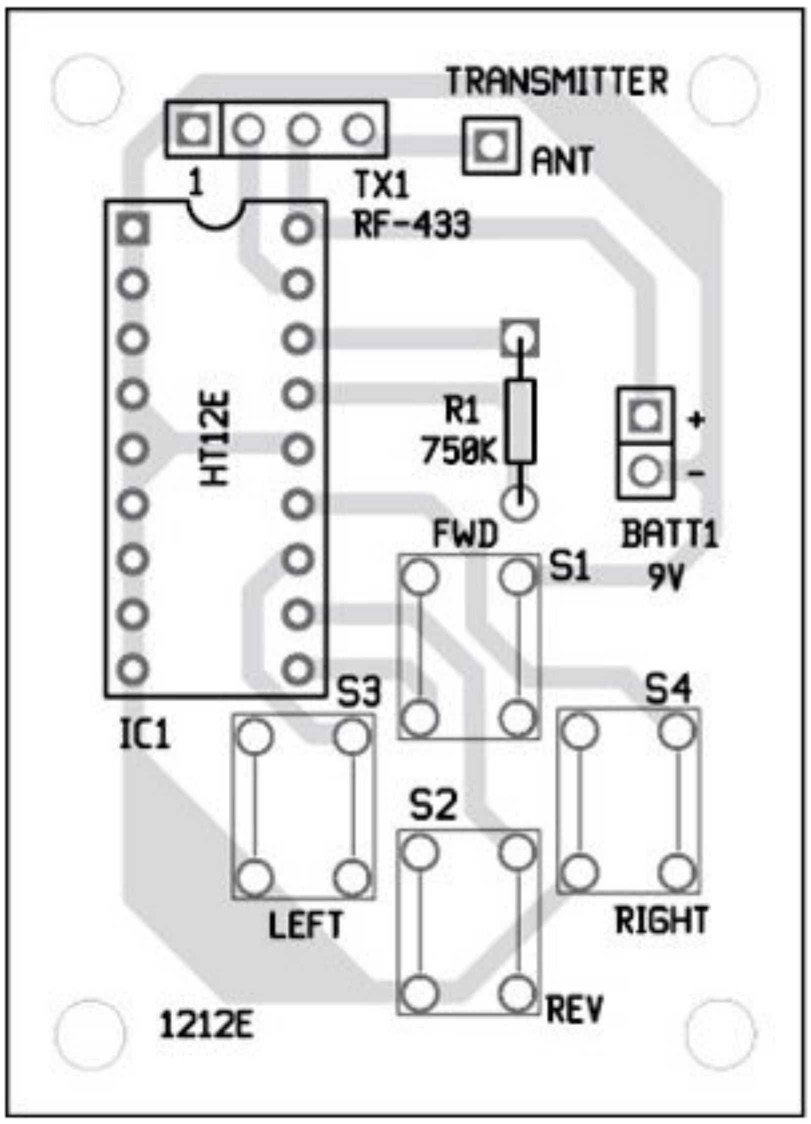


Fig. 5: Component layout for the PCB in Fig. 4

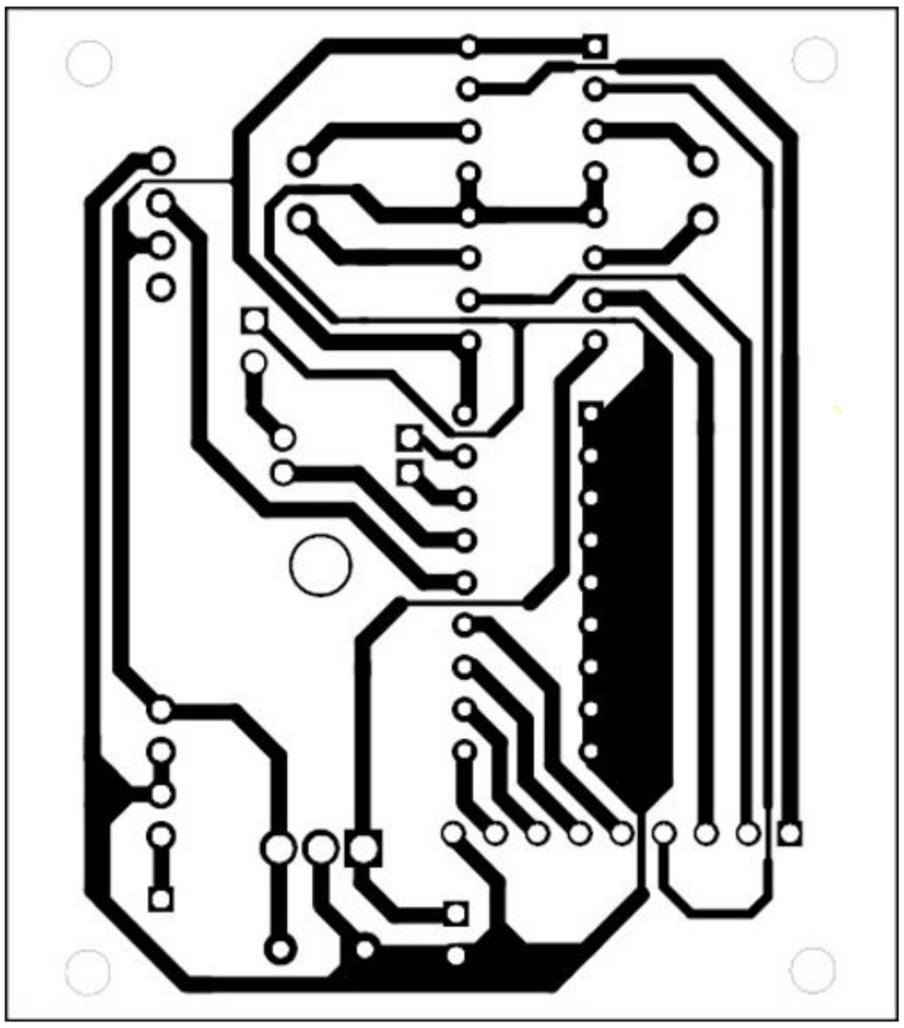


Fig. 6: A single-side PCB for the RF receiver

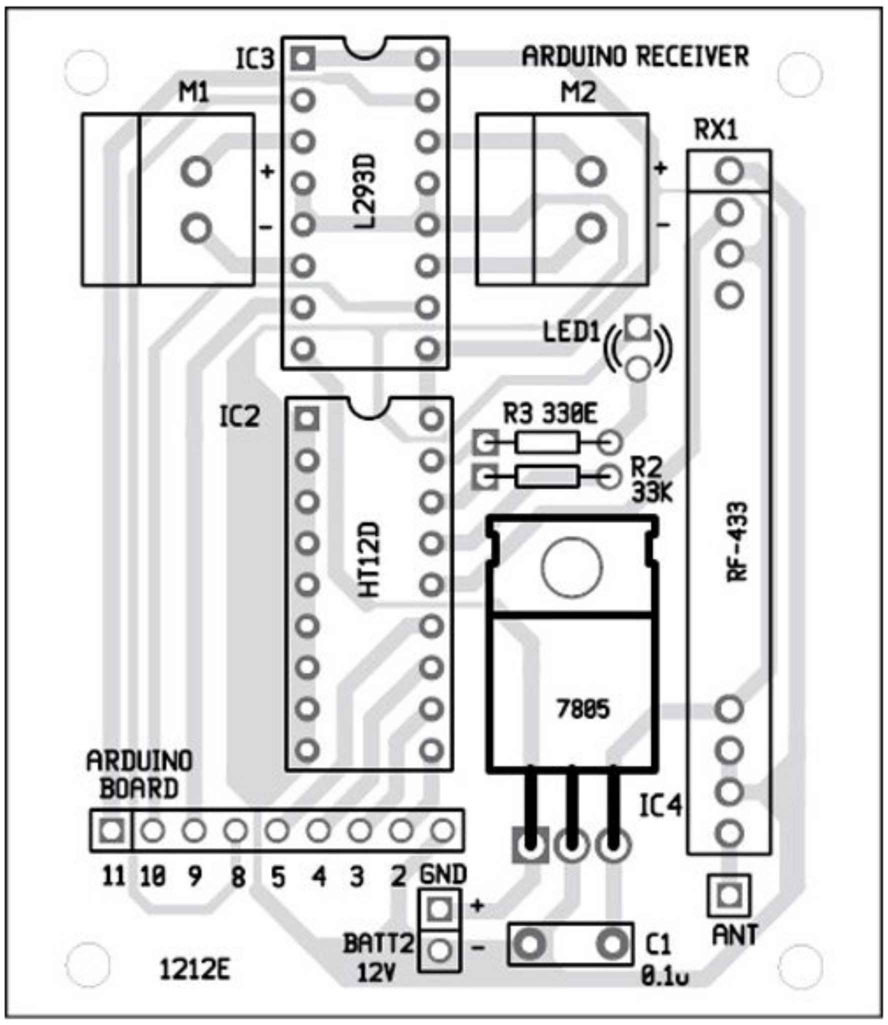


Fig. 7: Component layout for the PCB in Fig. 6

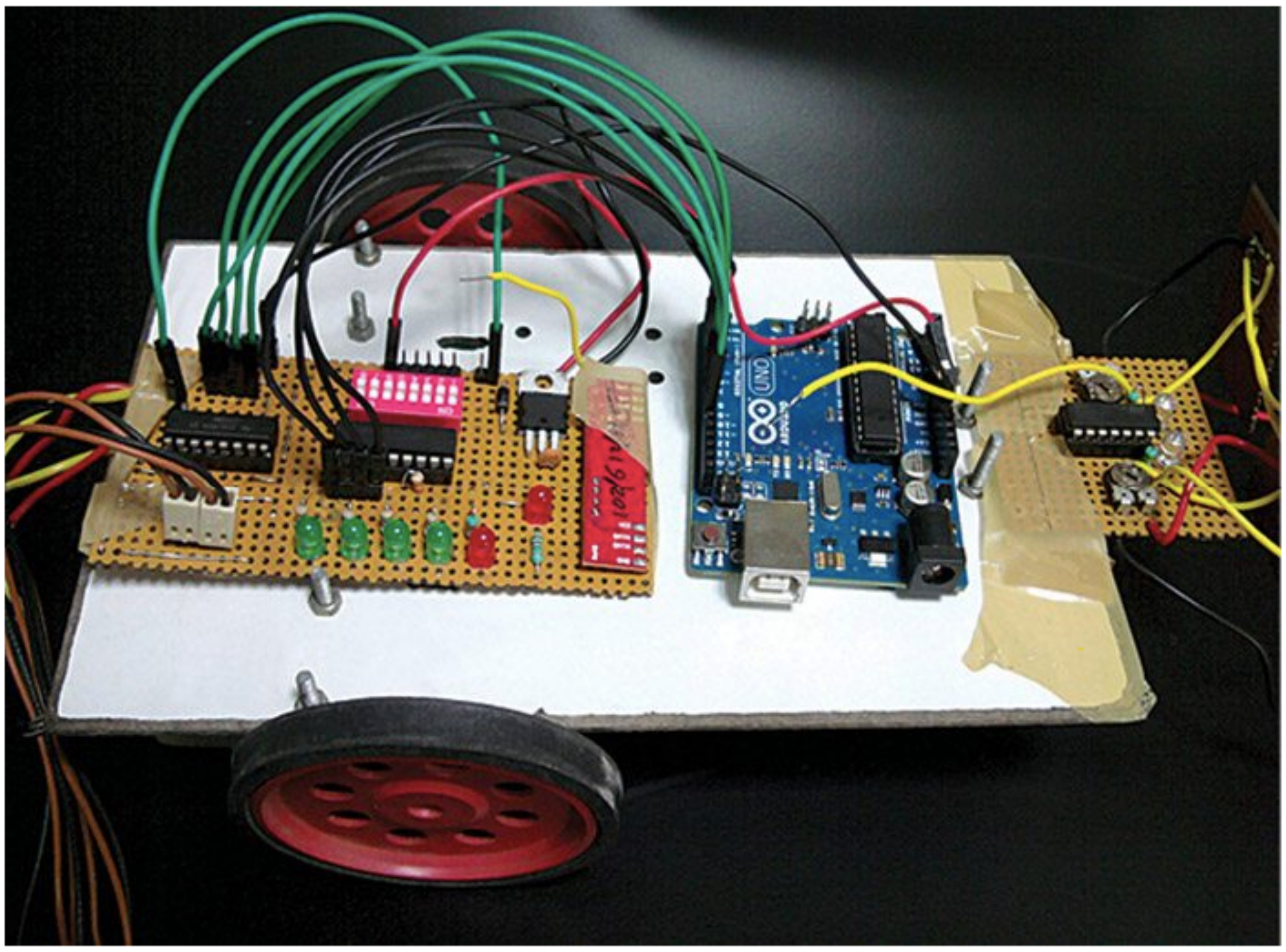


Fig. 8: Author's prototype of the Arduino based RF controlled robot

RFROBOT.INO

```
//RF Robot
int sw1 =2;
int sw2 =3;
int sw3 =4;
int sw4 =5;
int out1=8;
int out2=9;
int out3=10;
int out4=11;
void setup()
{
  pinMode(sw1,INPUT);
  pinMode(sw2,INPUT);
  pinMode(sw3,INPUT);
  pinMode(sw4,INPUT);
  pinMode(out1,OUTPUT);
  pinMode(out2,OUTPUT);
  pinMode(out3,OUTPUT);
  pinMode(out4,OUTPUT);
}
void loop()
{
     if  ((digitalRead(sw1)==LOW)
&&  (digitalRead(sw2)==HIGH)   &&
(digitalRead(sw3)==HIGH)   &&
(digitalRead(sw4)==HIGH))
  {
    fwd();
  }
    else  if  ((digitalRead(sw1)==HIGH)
&&   (digitalRead(sw2)==LOW)   &&
(digitalRead(sw3)==HIGH)   &&
(digitalRead(sw4)==HIGH))
  {
    bwk();
  }
    else  if  ((digitalRead(sw1)==HIGH)
&&    (digitalRead(sw2)==HIGH)
&&   (digitalRead(sw3)==LOW)   &&
(digitalRead(sw4)==HIGH))
  {
    lft();
  }
    else  if  ((digitalRead(sw1)==HIGH)
&&   (digitalRead(sw2)==HIGH)   &&
(digitalRead(sw3)==HIGH)   &&
(digitalRead(sw4)==LOW))
  {
    rgt();
  }
  else
  {  digitalWrite(out1,LOW);
     digitalWrite(out2,LOW);
     digitalWrite(out3,LOW);
     digitalWrite(out4,LOW);
  }
}
void fwd()
{  digitalWrite(out1,HIGH);
   digitalWrite(out2,LOW);
   digitalWrite(out3,HIGH);
   digitalWrite(out4,LOW);
}
void bwk()
{   digitalWrite(out1,LOW);
    digitalWrite(out2,HIGH);
    digitalWrite(out3,LOW);
    digitalWrite(out4,HIGH);
}
void lft()
{   digitalWrite(out1,LOW);
    digitalWrite(out2,HIGH);
    digitalWrite(out3,HIGH);
    digitalWrite(out4,LOW);
}

void rgt()
{   digitalWrite(out1,HIGH);
    digitalWrite(out2,LOW);
    digitalWrite(out3,LOW);
    digitalWrite(out4,HIGH);
}
```

Download PCB and component layout PDFs: click here

Download source code: click here

31

LED SCROLLING DISPLAY

BY HARWINDER SINGH

A light-dependent resistor (LDR) whose resistance is inversely proportional to the intensity of light is often used as a sensor in electronic projects that involve the use of light. This project uses an LDR to control the speed of a DC motor.

Most outdoor LED displays and some indoor LED displays are built around individually mounted LEDs. Presented here is a LED scrolling display that uses 64

LEDs to display alphabets and numbers. A cluster of red, green and blue diodes is driven together to form a full-colour display.

In a dot-matrix LED display, the LEDs are wired together in rows and columns to minimise the number of pins required to drive them. For example, an 8×8 matrix of LEDs, shown in Fig. 1, would need 64 I/O pins—one for each LED pixel. By wiring all anodes together in rows (R1 through R8) and cathodes in columns (C1 through C8), the required number of I/O pins is reduced to 16.

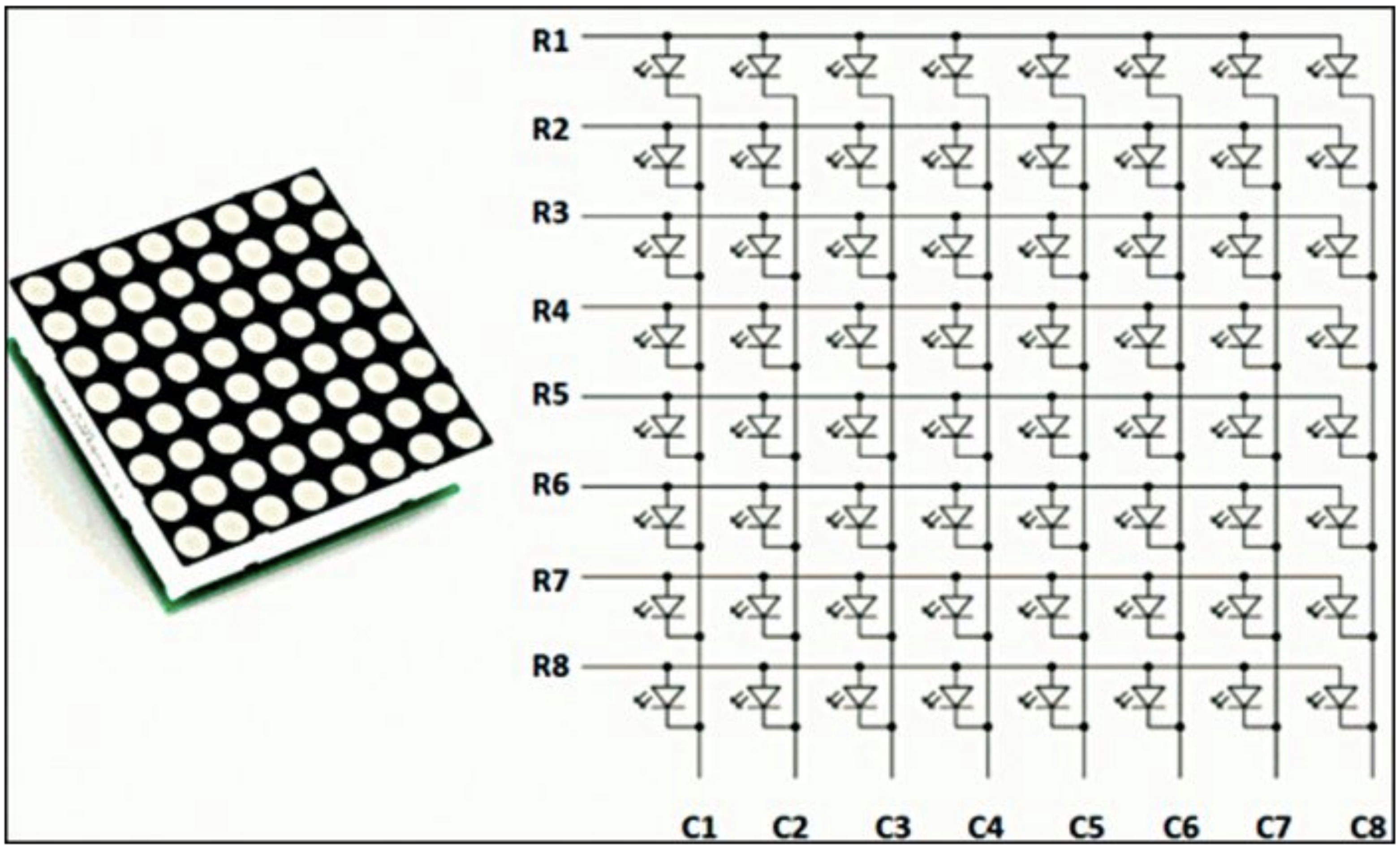


Fig. 1: Structure of an 8×8 LED dot-matrix display

Each LED is addressed by its row and column number. In Fig. 1, if R4 is pulled high and C3 is pulled low, the LED in the fourth row and third column will turn on. Alphabets and numerals can be displayed by fast scanning of either rows or columns. In this project, column scanning has been used.

Working of a dot-matrix display

Fig. 2 shows which LEDs in a 5×7 matrix of LEDs are to be turned on to display the English alphabet A. The seven rows and five columns of the array are controlled through a microcontroller.

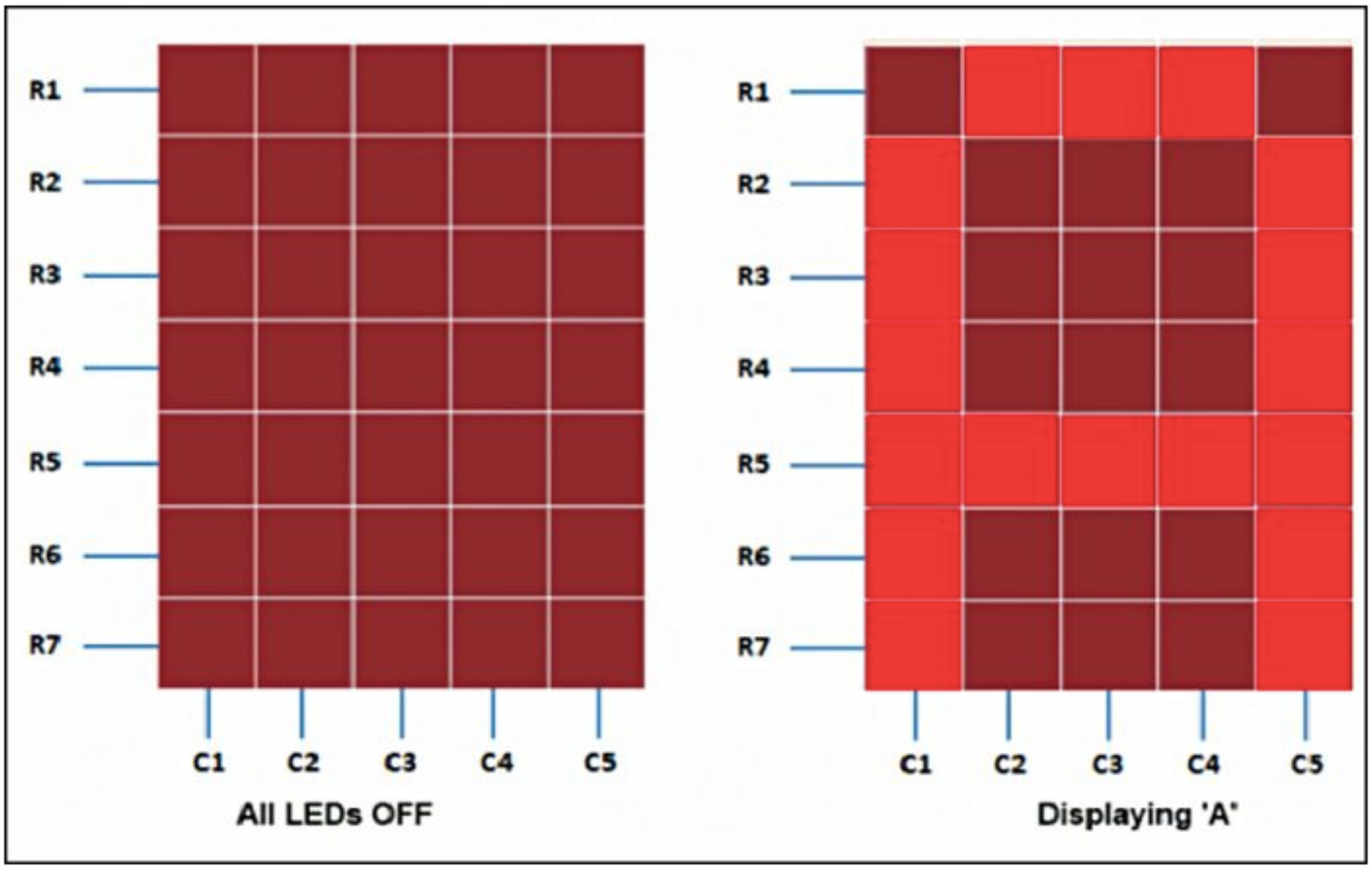


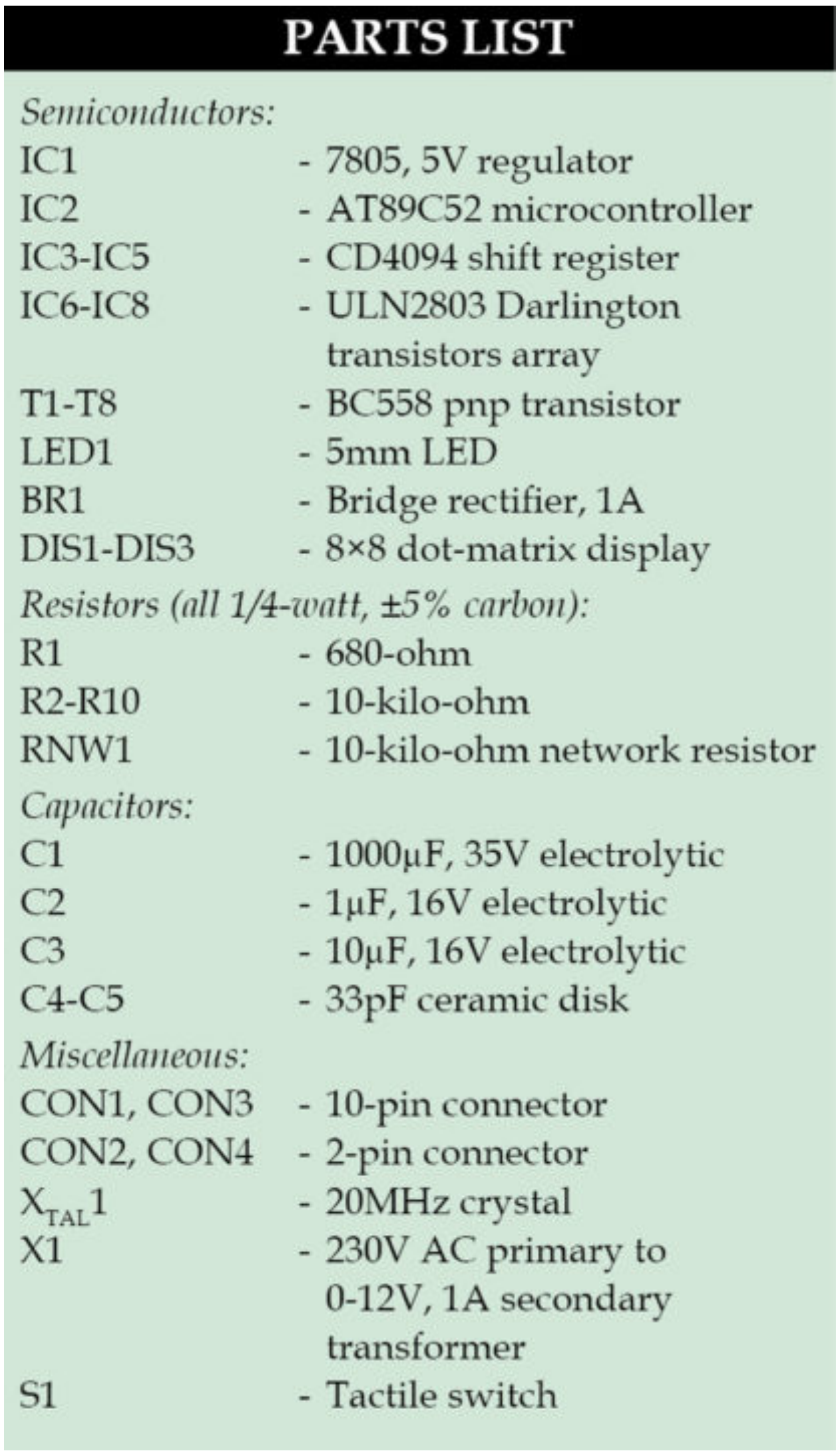
Fig. 2: 5×7 array of LEDs

If we want to display alphabet A, we will first select column C1 (which means C1 is pulled low in this case) and deselect other columns by blocking their ground paths (one way of doing that is by pulling C2 through C5 pins to logic high). Now, the first column is active, and you need to turn on the LEDs in rows R2 through R7 of this column, which can be done by applying forward-bias voltages to these rows.

Next, select column C2 (and deselect all other columns) and apply forward-bias voltages to resistors R1 and R5, and similarly for columns C3 and C4. Then, activate column C5 by pulling it down and deselect other columns, and apply forward-bias voltages to LEDs in rows R2 through R7.

By repeating these steps quickly (>100 times per second), and turning on the respective LEDs in each row of that column, the persistence of vision comes into play and we perceive the displayed image of the alphabet A as still.

You must have noticed that across each row one pin is sourcing the current for only one LED at a time, but a column pin may have to sink the currents from more than one LED. For example, column C1 should be able to sink the current from six LEDs while displaying alphabet A. A microcontroller has low sourcing as well as sinking capabilities. To obviate this limitation, external transistor arrays or buffers are used. In this project, PNP transistor BC558 (T1-T8) has been used for this purpose. In the circuit, an 8×8 dot-matrix has been used.



PARTS LIST

Semiconductors:	
IC1	- 7805, 5V regulator
IC2	- AT89C52 microcontroller
IC3-IC5	- CD4094 shift register
IC6-IC8	- ULN2803 Darlington transistors array
T1-T8	- BC558 pnp transistor
LED1	- 5mm LED
BR1	- Bridge rectifier, 1A
DIS1-DIS3	- 8×8 dot-matrix display
Resistors (all 1/4-watt, ±5% carbon):	
R1	- 680-ohm
R2-R10	- 10-kilo-ohm
RNW1	- 10-kilo-ohm network resistor
Capacitors:	
C1	- 1000µF, 35V electrolytic
C2	- 1µF, 16V electrolytic
C3	- 10µF, 16V electrolytic
C4-C5	- 33pF ceramic disk
Miscellaneous:	
CON1, CON3	- 10-pin connector
CON2, CON4	- 2-pin connector
$X_{TAL}1$	- 20MHz crystal
X1	- 230V AC primary to 0-12V, 1A secondary transformer
S1	- Tactile switch

A brief description of the ICs used

IC1 is a 7805, 5V regulator IC, which provides the 5V output voltage for driving the circuit around microcontroller IC2. AT89C52 (IC2) is a low-power, high-performance CMOS 8-bit microcontroller. It has 8k bytes of Flash ROM, 256 bytes of RAM, 32 I/O lines, three 16-bit timers/counters, a six-vector two-level interrupt, a full duplex serial port, an on-chip oscillator and on-clock circuitry.

CD4094 (IC3-IC5) is an 8-stage serial shift register, having a storage latch associated with each stage for stroking data from the serial input to parallel buffered 3-state outputs. The parallel outputs may be connected directly to common bus lines. Data is shifted on positive clock transition. Data in each shift register stage is transferred to the storage register when the strobe input is high. Data in the storage register appears at the outputs, whenever the Output-Enable signal is high. Two serial outputs are available for cascading a number of CD4094 devices.

Data is available at the Q serial output terminals on positive clock edges to allow for high-speed operation in cascaded system. The same serial information, available at the Q terminals on the next negative clock edge, provides a means for cascading CD4094 devices when the clock rise time is slow.

ULN2803 (IC6–IC8) is an octal high-voltage, high-current Darlington transistor array. The eight NPN Darlington connected transistors (T1-T8) are ideally suited for interfacing between low-logic-level digital circuitry and the higher current/voltage requirements of lamps and relays. The device features open-collector outputs and freewheeling clamp diodes for transient suppression. It is designed to be compatible with standard TTL families of Ics.

LED scrolling display circuit and working

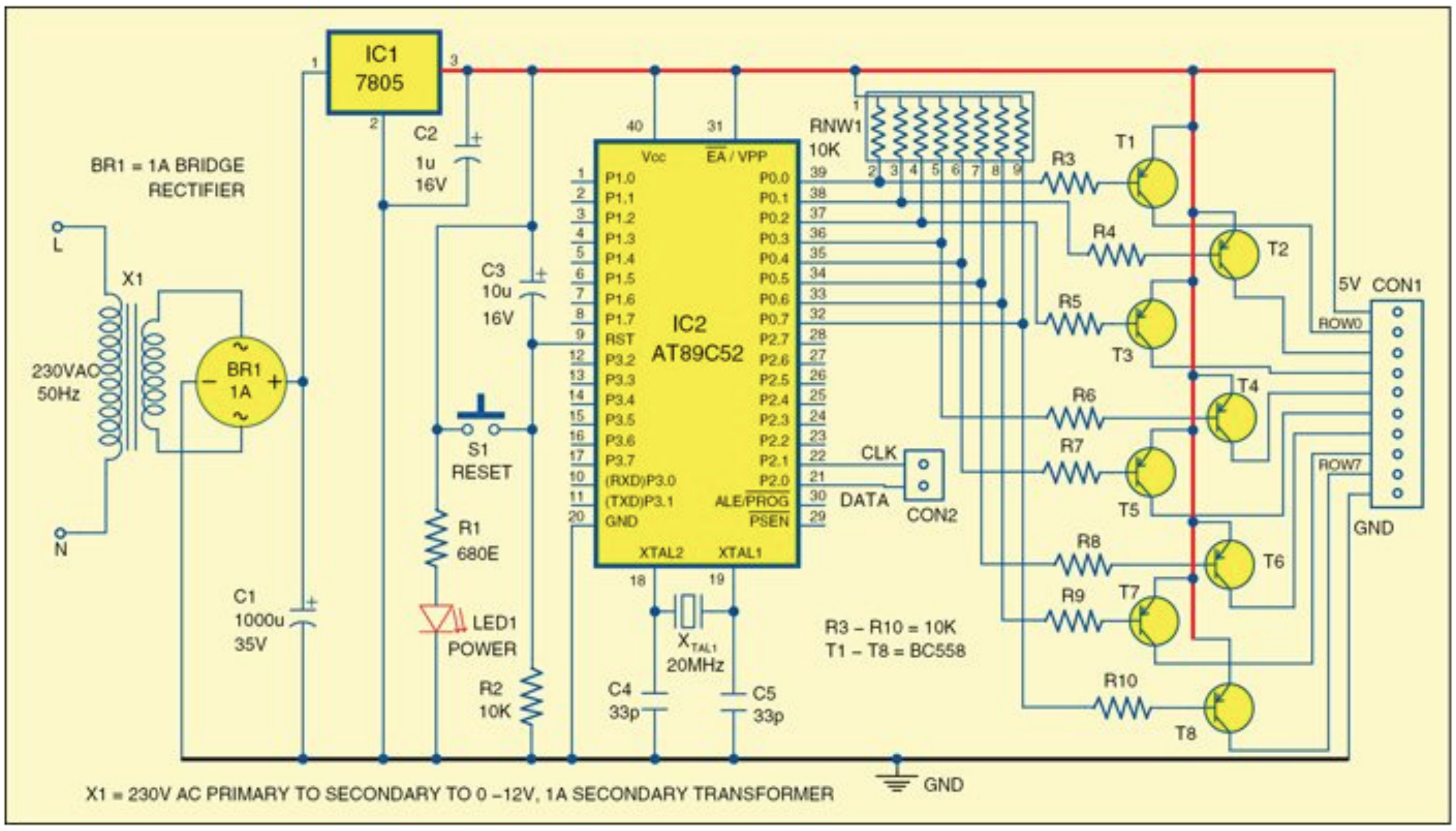


Fig. 3: LED Scrolling Display: Power supply and controller circuit

The circuit diagram is divided here into two parts. The first part has a power supply and controller circuit as shown in Fig. 3. The display unit is in the second part as shown in Fig. 4.

The power supply circuit is built around a step-down transformer, bridge rectifier and 5V regulator. The configuration is conventional. The circuit provides regulated 5V for operation of the circuit.

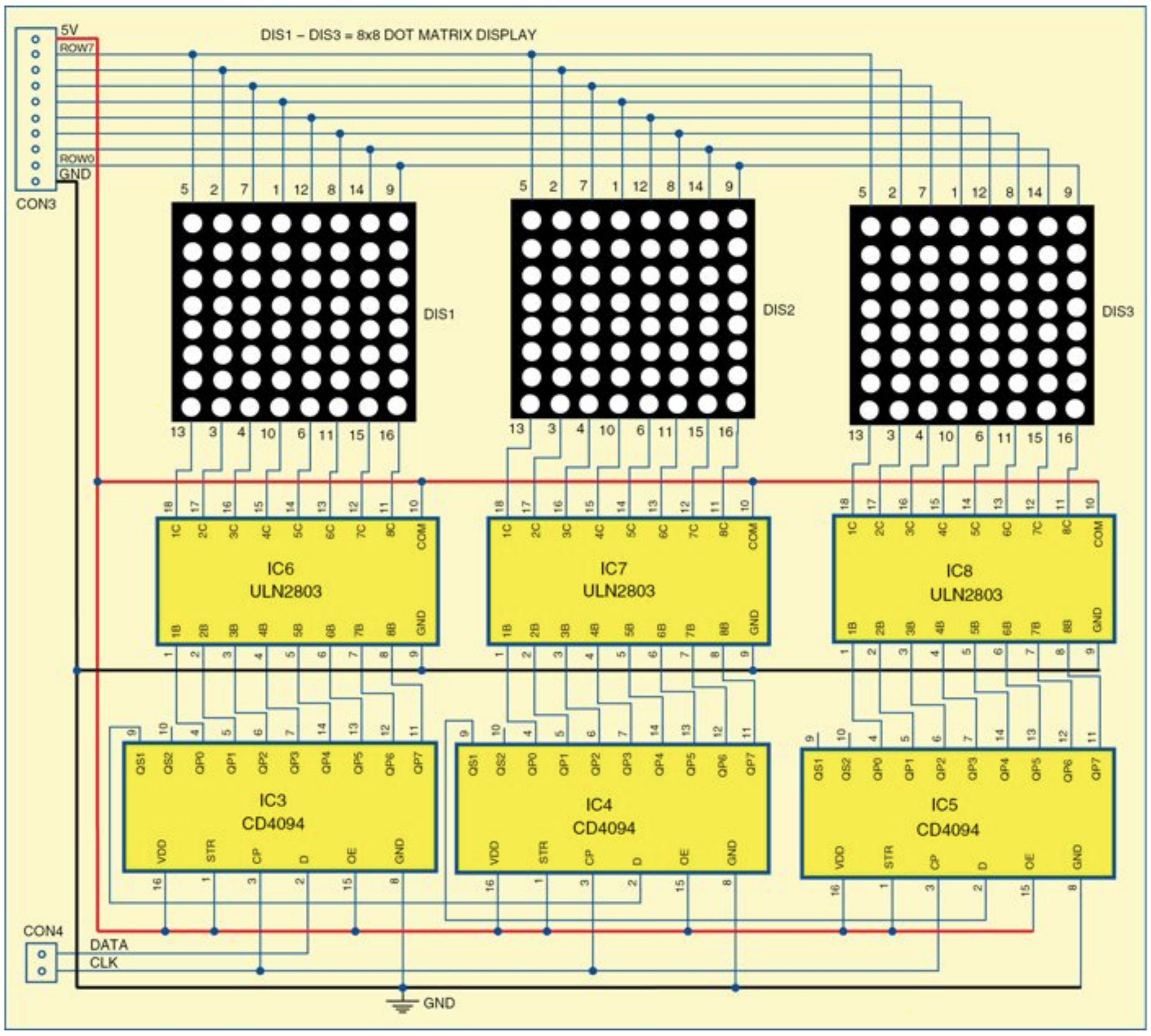


Fig. 4: LED Scrolling Display: Display unit

Microcontroller AT89C52 provides outputs to control the logic levels for the eight rows R0-R7 through port A (P0.0-P0.7). RNW1 is a network resistor, which acts as a pull-up resistor for port 0. An array of eight pnp BC558 transistors (T1-T8) working as current drivers takes care of the current required for LEDs of the dot-matrix; the microcontroller has low sourcing capabilities. The reset arrangement is made at pin 9 of the microcontroller with the help of capacitor C3, resistor R2 and switch S1.

The circuit for the display unit is shown in Fig. 4. It is built around three CD4094 cascaded 8-bit shift registers (IC3-IC5), three ULN2803 high-current Darlington transistor arrays (IC6-IC8) and three 8×8 dot-matrix displays (DIS1-DIS3). Clock and data pulses are generated by the microcontroller at ports P2.1 and P2.0, respectively. These are transferred to the display unit through connectors CON2 and CON4.

Clock pulses are fed to pin 3 of all the three shift registers, while data pulses are fed to pin 2 of the first shift register, IC3. The output of the first shift register from pin 9 is fed to pin 2 of the second shift register, IC4. To complete the cascading, output of the second shift register from its pin 9 is fed to pin 2 of the third shift register, IC5.

Data from the shift registers is used for activation or deactivation of columns of the three dot-matrix displays (DIS1-DIS3) through IC6-IC8 ICs. For controlling the logic levels at rows R0-R7, data from port P0 is transferred from the microcontroller to the display unit through connectors CON1 and CON3, with a supply of 5V and common ground to display unit of these two connectors.

Software

The source program for the LED scrolling display of 'EFY INDIA' is written in Assembly language and compiled using Keil µVision4 compiler. The generated hex code is burnt into the microcontroller using a suitable programmer.

Construction and testing of LED scrolling display

A single-layer PCB layout for the power supply and the microcontroller unit is shown in Fig. 5 and its component layout in Fig. 6. PCB for the display unit is a double-sided one. Track layout of its bottom layer is shown in Fig. 7 and of top layer in Fig. 8. Component

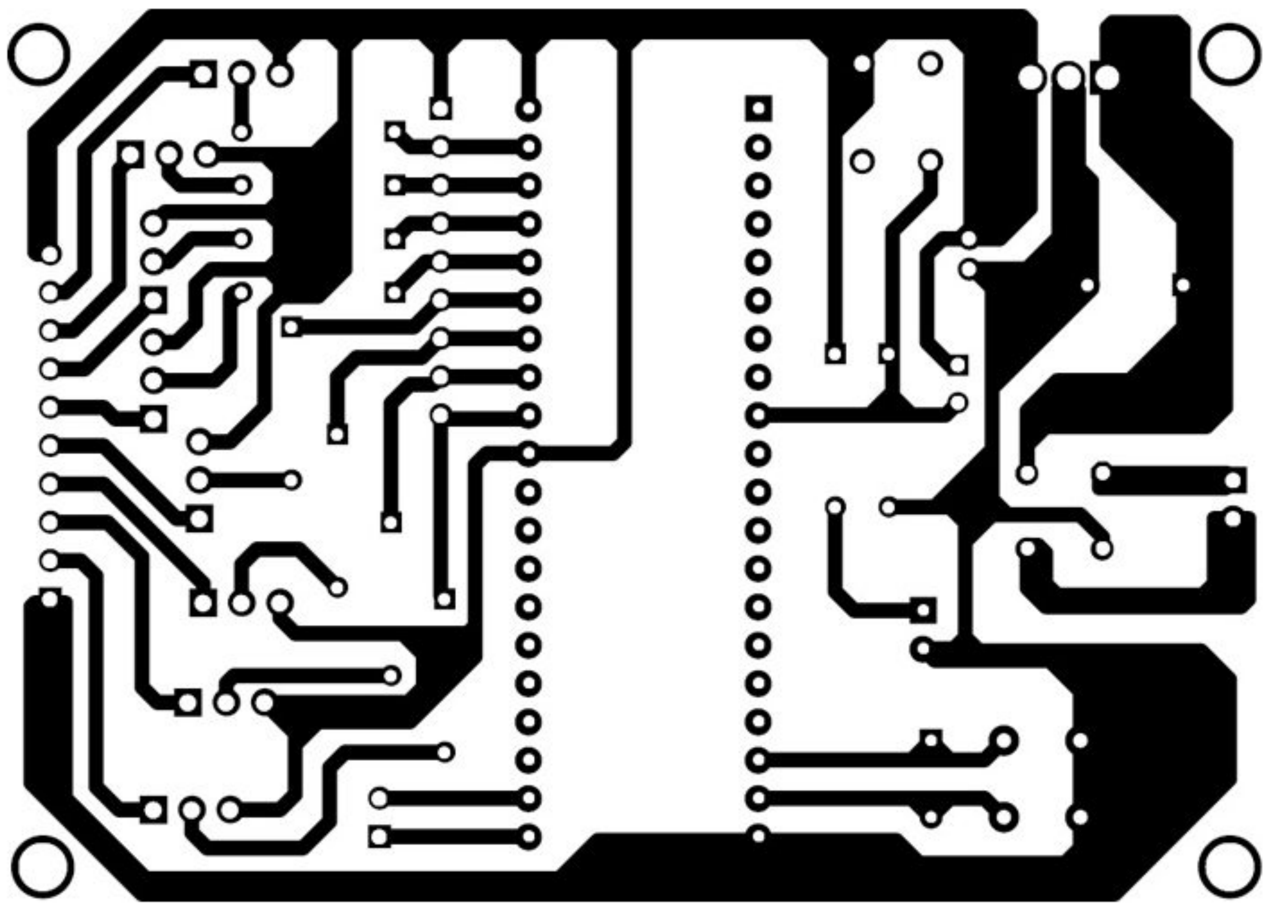


Fig. 5: A PCB layout for the power supply and microcontroller unit

and track layouts of the top layer as well as the bottom layer are shown in Fig. 9.

Assemble the circuits on the PCBs to save time and avoid assembly errors.

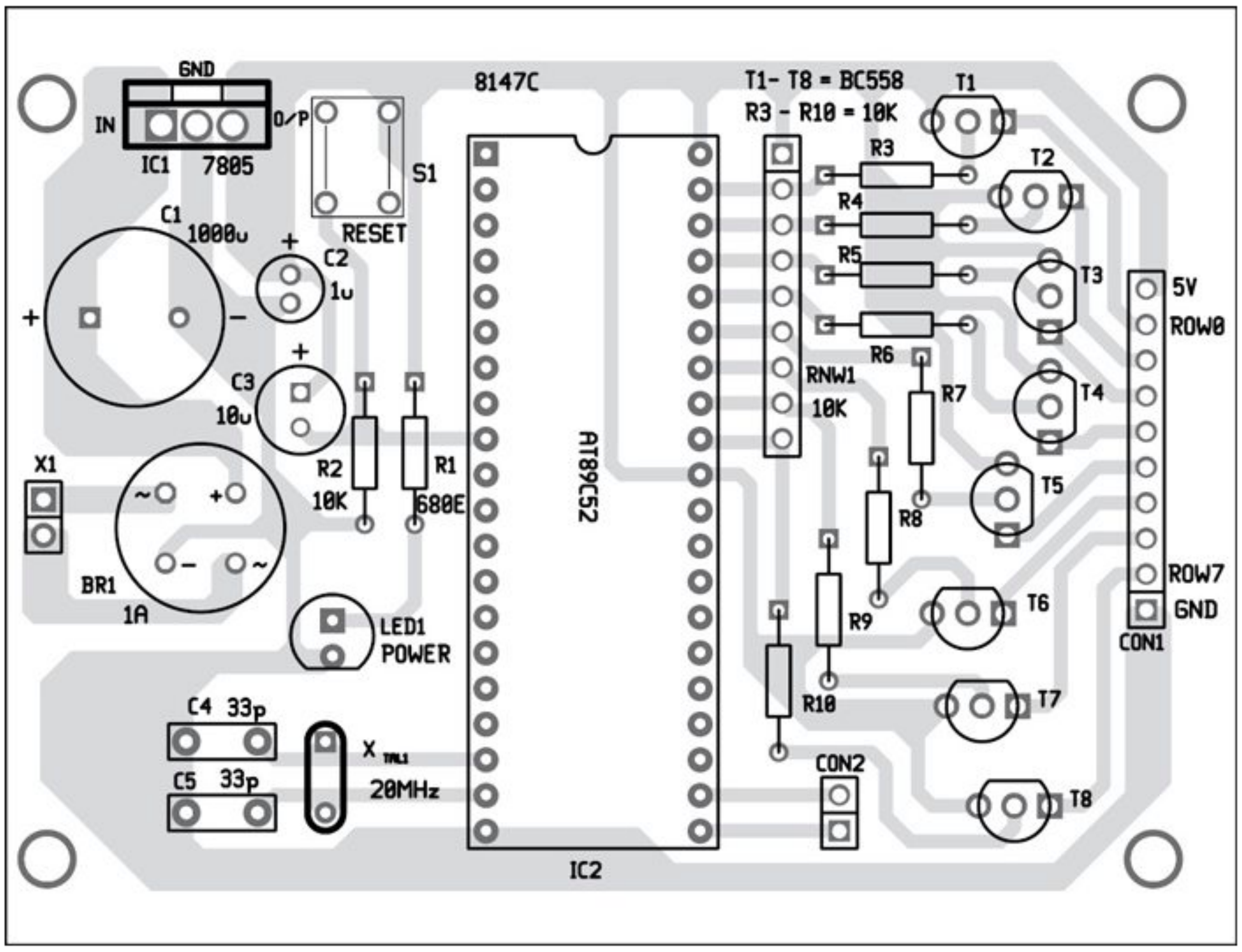


Fig. 6: Component layout of power supply and microcontroller unit

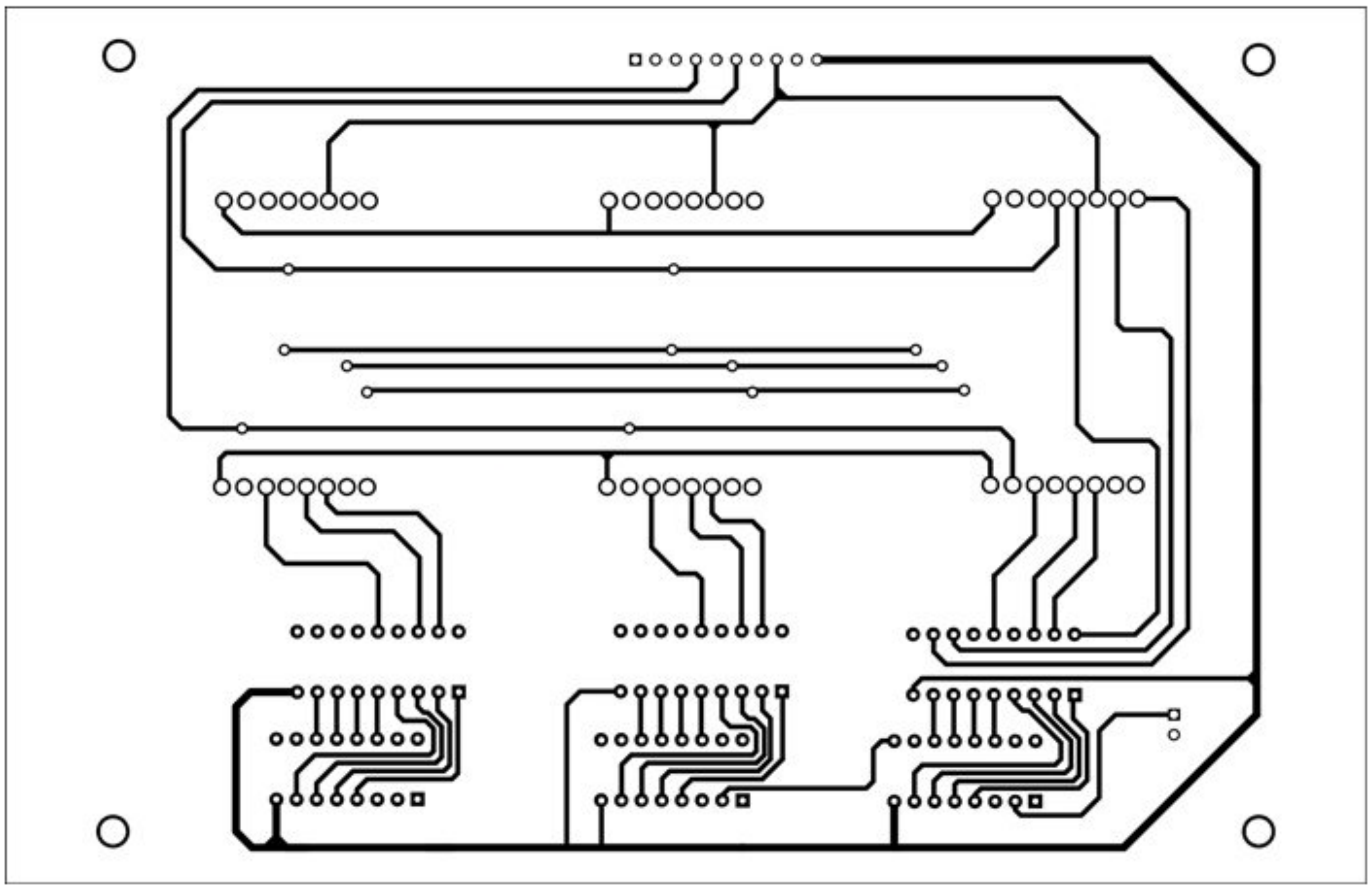


Fig. 7: Track layout of the bottom layer of display unit

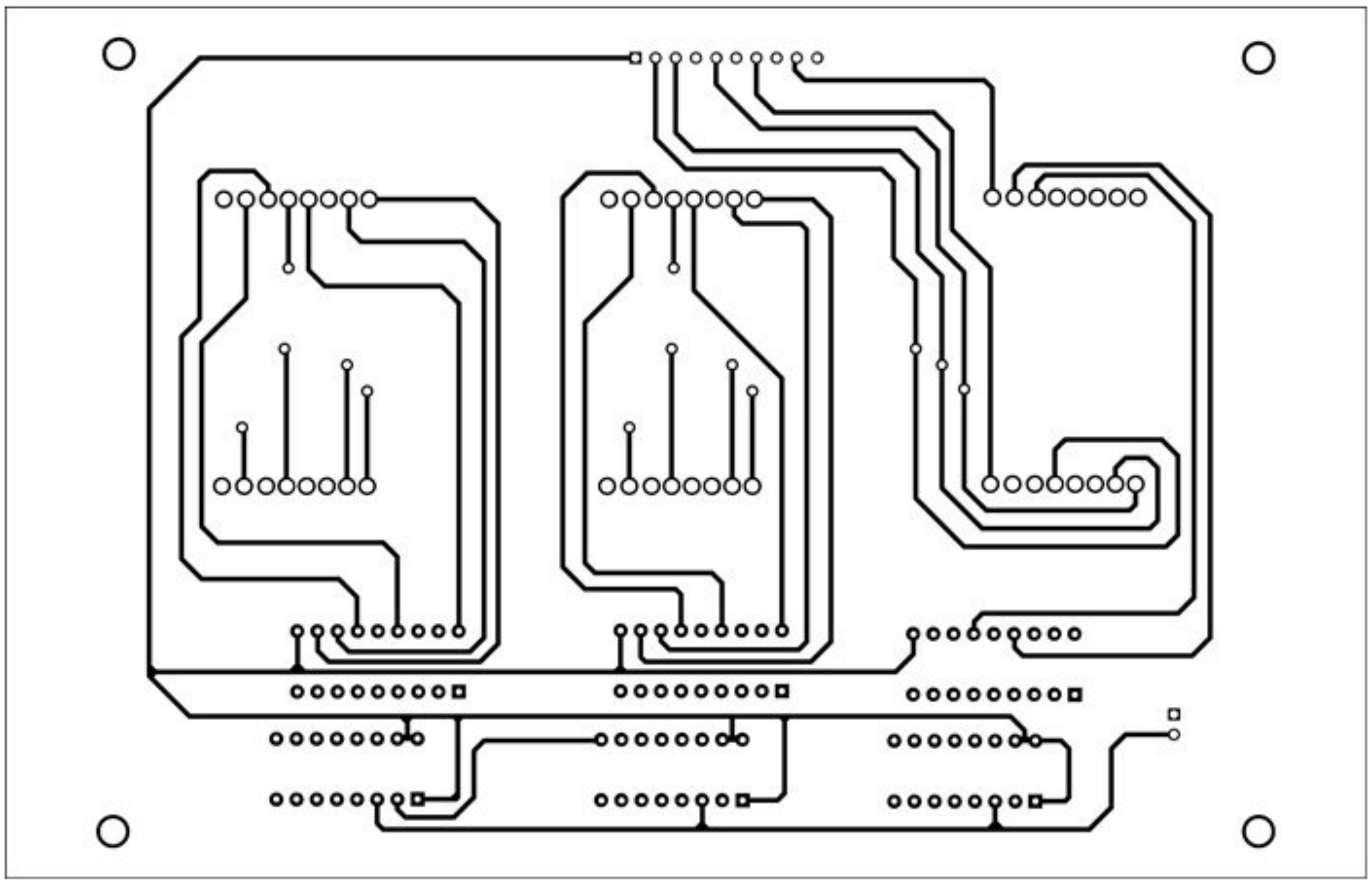


Fig. 8: Track layout of the top layer of display unit

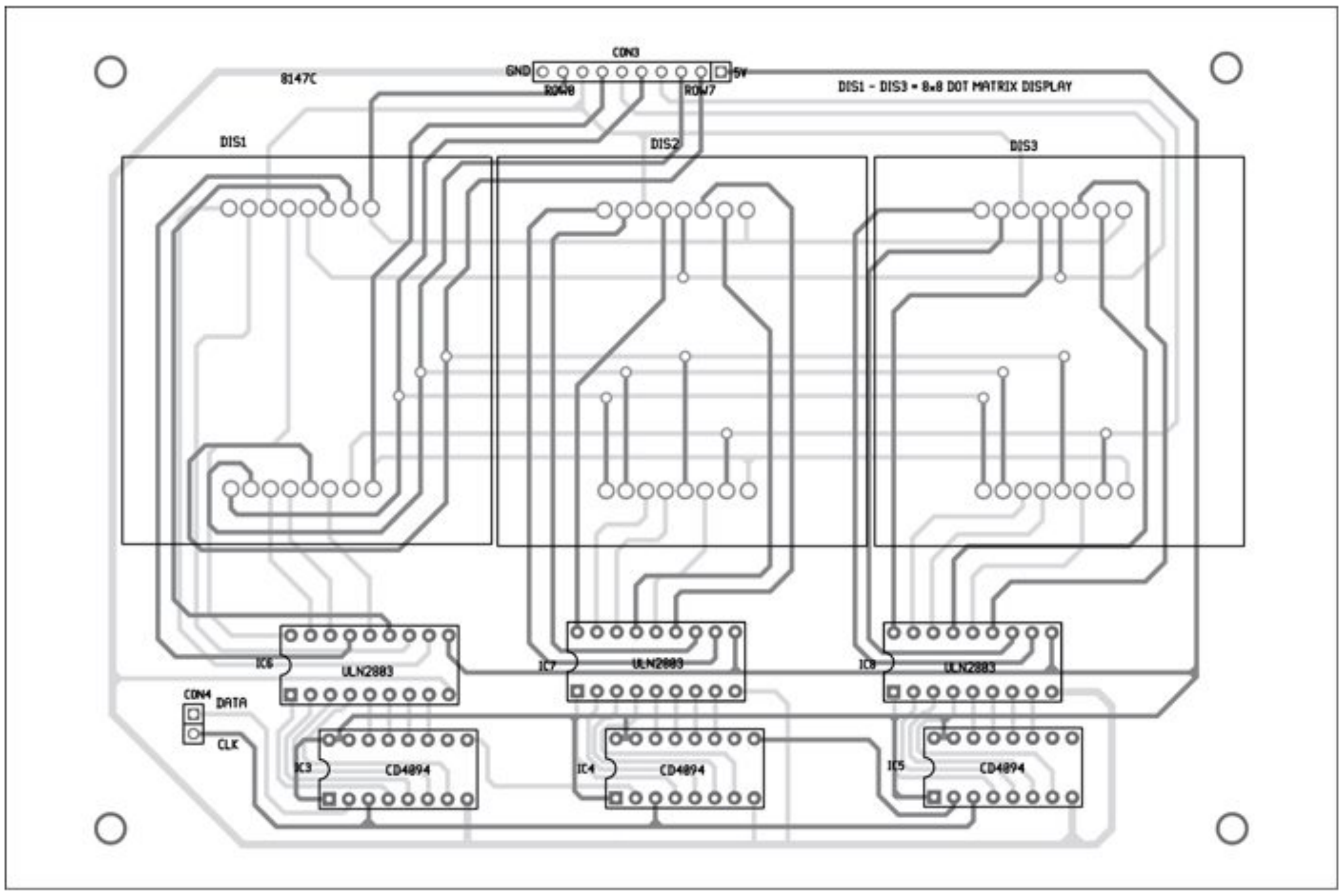
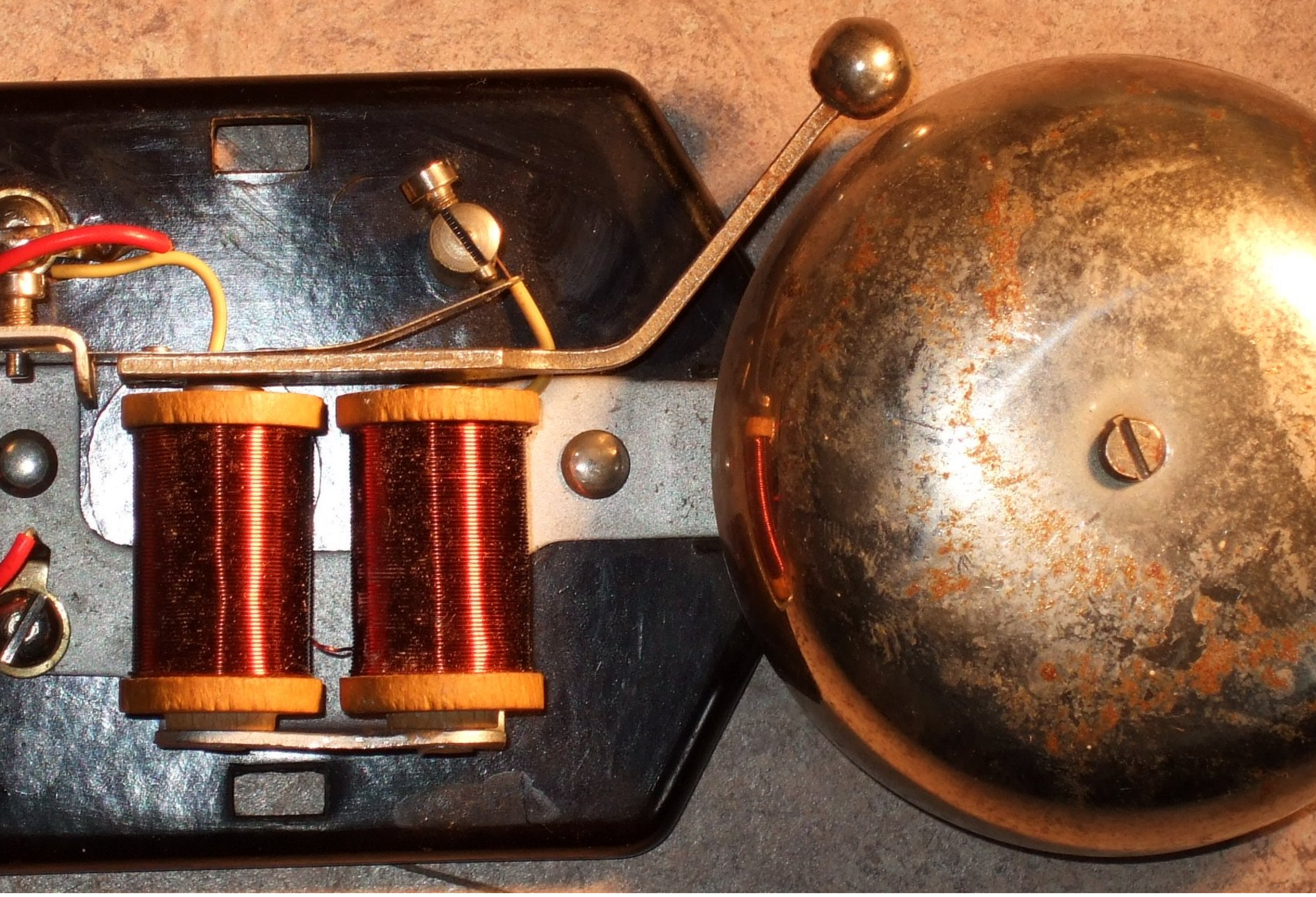


Fig. 9: Component-side track layouts of top as well as bottom layers of the display unit

Download PCB and component layout PDFs: click here

Download source code: click here



32

SMART TOUCH DISPLAY MULTI-DOOR DOORBELL SYSTEM

BY ASHWINI KUMAR SINHA

Technology is advancing daily and everything is turning smarter. A number of traditional devices that we used at our homes are becoming smarter now, including the doorbells which are also evolving with new features and styles. Today you will find many fancy doorbells in the market with touch switch system, but they are very costly to set up and everybody can't afford to have it.

So, we have decided to help you make your own DIY touch display doorbell. This doorbell has a lot of unique systems, which will give you a new experience altogether. Another interesting feature of this doorbell is that it has separate ringing bells for each of the room in your house or office. It also has a special notification system to let the visitors know whether the office is closed or not, or if any of the family members is available or not.

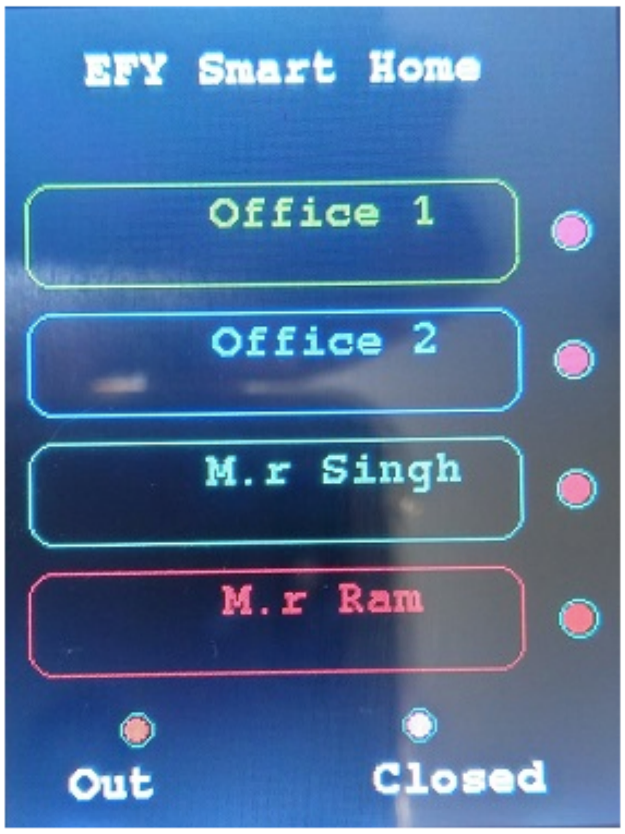


Fig 1.Prototype touch screen

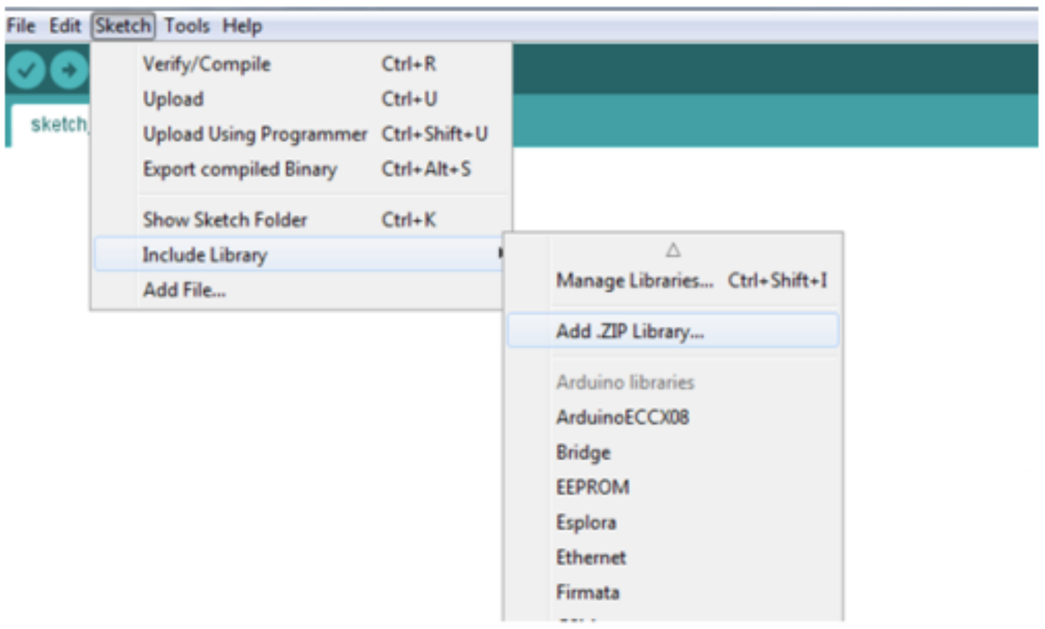
So, let's start the project.

Bill of Material

Component Name	Quantity	Description	Cost Approx. In INR
Arduino Uno	1	For Programing	350
2.4 TFT Display	1	MCU friend St7789v	600
Bluetooth HC 05	1	Optional For setting status	300
Wires	20cm	For Connection	10
Buzzer	4	For Bells	20*4
Total Cost			**1340**

Setting Arduino Library

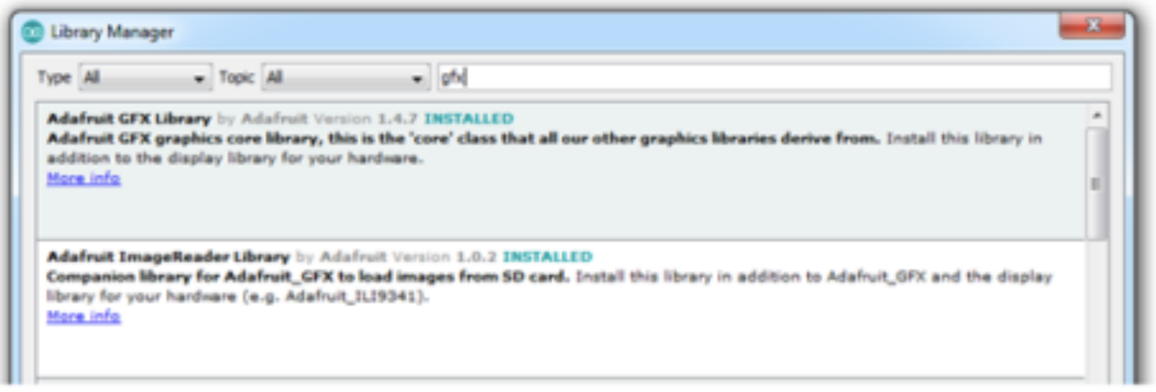
First, we will to install the required library to Arduino IDE. For this, Go to Sketch menu and click on manage library. Then search the library name and click on install. The library we need in this project are Addafruit GFX library and Touch screen library.


File Edit Sketch Tools Help
sketch
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
ArduinoECCX08
Bridge
EEPROM
Esplora
Ethernet
Firmata
Fig 3. Opening Library manager.

NOTE: Every touch screen has their own library. So, find a library according to your TFT screen model and install it. Here we have used MCUFriend 2.4 TFT LCD Library St7789v and installed it in our Arduino IDE. Refer to (Fig 1,2,3,4) .

Coding

First, we will initialise the libraries in code and then declare the required variables (home1, home 2, home 3, home 4) that we are going to use for separate rooms. After that we will add fonts and colour code for TFT Display (Refer Fig 6).

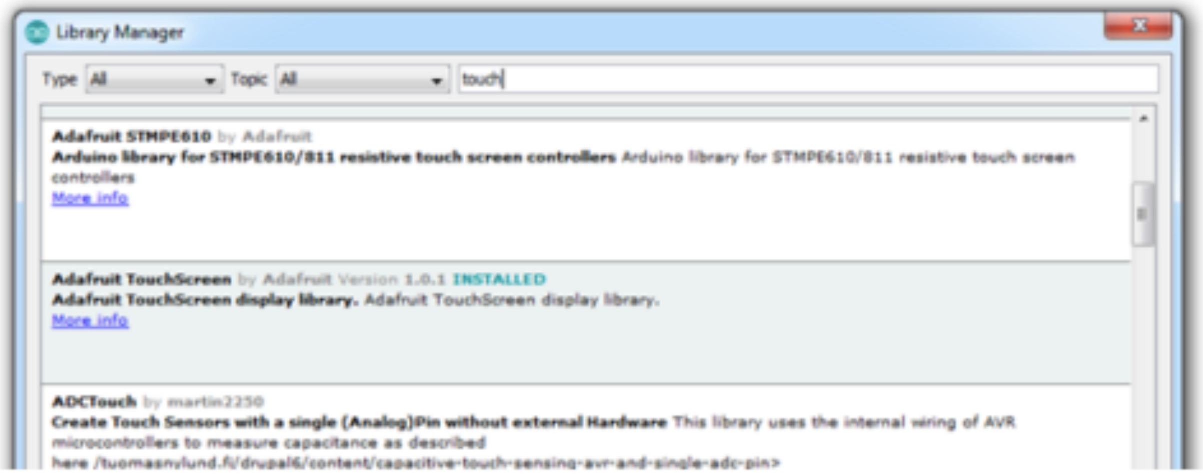

Library Manager

Type: All | Topic: All | gfx

Adafruit GFX Library by Adafruit Version 1.4.7 INSTALLED
Adafruit GFX graphics core library, this is the 'core' class that all our other graphics libraries derive from. Install this library in addition to the display library for your hardware.
More info

Adafruit ImageReader Library by Adafruit Version 1.0.2 INSTALLED
Companion library for Adafruit_GFX to load images from SD card. Install this library in addition to Adafruit_GFX and the display library for your hardware (e.g. Adafruit_ILI9341).
More info

Fig 4. Installing Adafurit GFX library


Library Manager

Type: All | Topic: All | touch

Adafruit STMPE610 by Adafruit
Arduino library for STMPE610/811 resistive touch screen controllers Arduino library for STMPE610/811 resistive touch screen controllers
More info

Adafruit TouchScreen by Adafruit Version 1.0.1 INSTALLED
Adafruit TouchScreen display library. Adafruit TouchScreen display library.
More info

ADCTouch by martin2250
Create Touch Sensors with a single (Analog)Pin without external Hardware This library uses the internal wiring of AVR microcontrollers to measure capacitance as described here /tuomasnylund.fi/drupal6/content/capacitive-touch-sensing-avr-and-single-adc-pin>

Fig 5. Installing touch screen library

Next, we need to create a function to get the touch points. And in this function, we have to set the range of pressure to enable the touch screen display to read the touch points. After that we will map the values with the TFT Display's width and height to identify the touch points. (Refer fig 7)

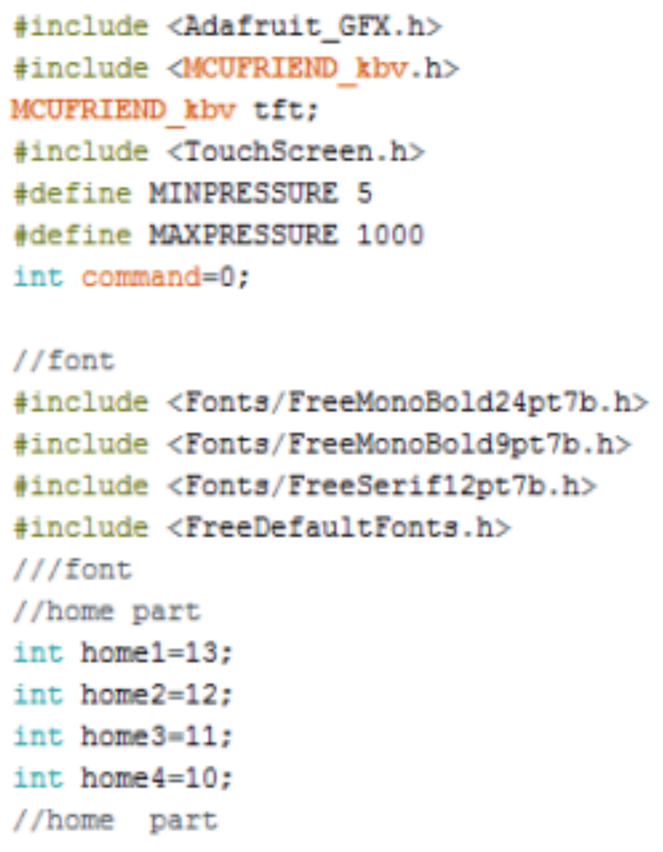

```
#include <Adafruit_GFX.h>
#include <MCUFRIEND_kbv.h>
MCUFRIEND_kbv tft;
#include <TouchScreen.h>
#define MINPRESSURE 5
#define MAXPRESSURE 1000
int command=0;

//font
#include <Fonts/FreeMonoBold24pt7b.h>
#include <Fonts/FreeMonoBold9pt7b.h>
#include <Fonts/FreeSerif12pt7b.h>
#include <FreeDefaultFonts.h>
///font
//home part
int home1=13;
int home2=12;
int home3=11;
int home4=10;
//home  part
```

Fig 6. Arduino code

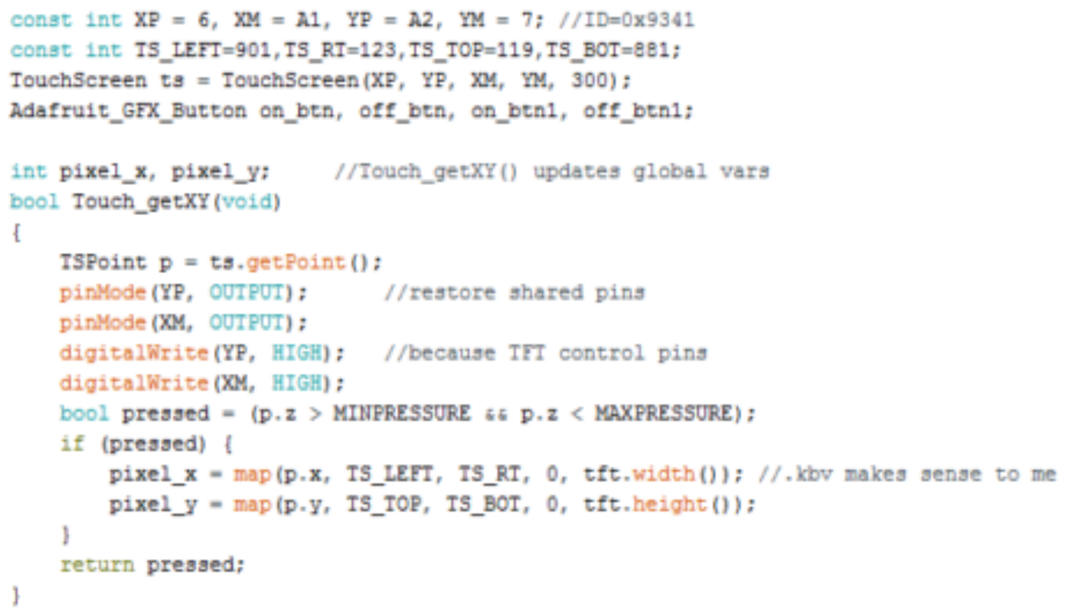
In the next step, we will create a setup function to declare the pin modes of Arduino and then create buttons that we want to display on the screen of Touch Display module. In the below code snippet we have created four buttons for four rooms. Along with buttons, we have also created a few notification dots that show the working status of the bell. The colour of these notification dots will change according to the status (for example, it will turn red when the room is close and maroon when it is open). (Refer fig 8)

Now, we will create a function to check and return the status of the buttons, that is weather the touch buttons are pressed or not. (Refer fig 9).

Then in loop function, we will create several 'if conditions' that will check the status of the buttons and perform the tasks according to code. (Refer fig 10.)

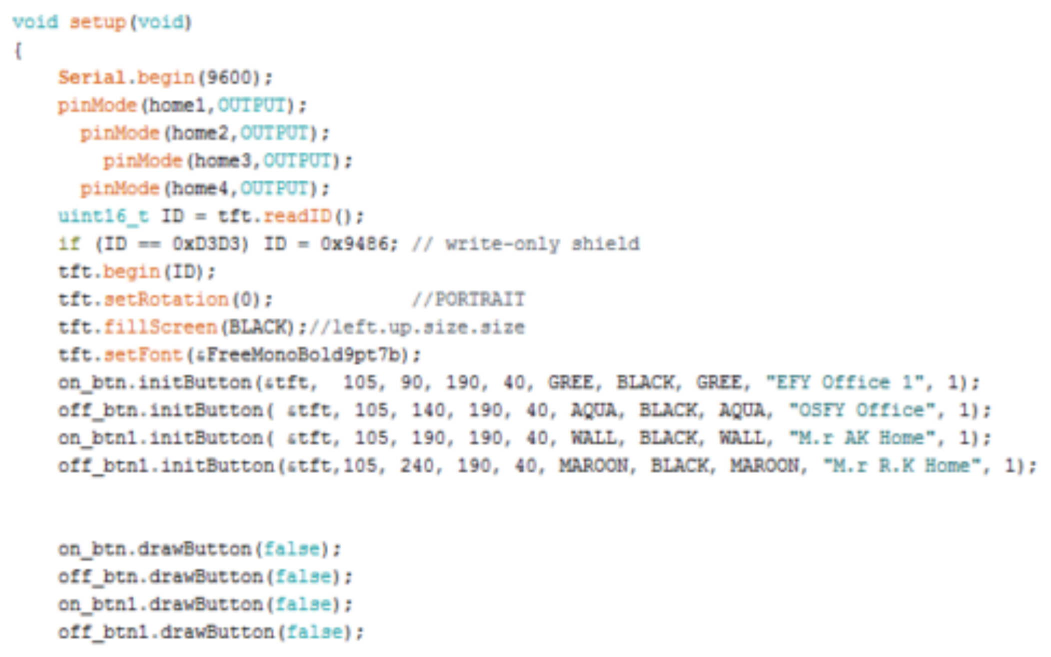
Testing

After connecting the components, crosscheck the wirings and connections. If all the connections are ok, then power the Arduino with 5-12 volt DC (You can use AC to DC adapter like cell phone



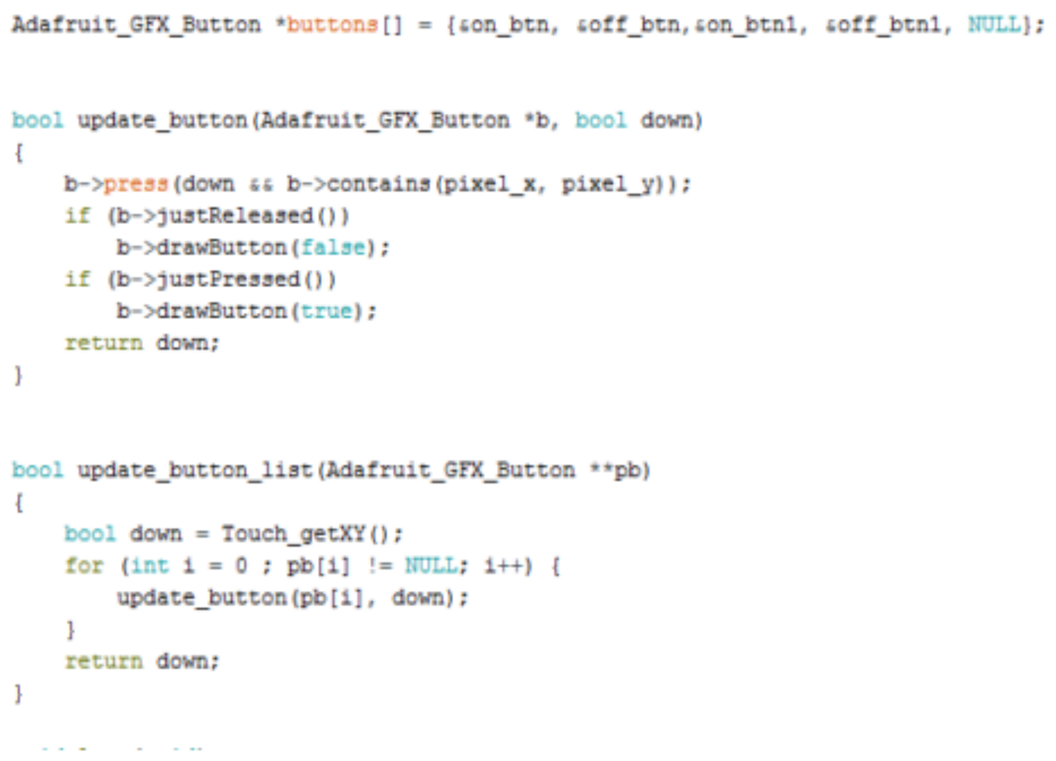
```
const int XP = 6, XM = A1, YP = A2, YM = 7; //ID=0x9341
const int TS_LEFT=901,TS_RT=123,TS_TOP=119,TS_BOT=881;
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);
Adafruit_GFX_Button on_btn, off_btn, on_btn1, off_btn1;

int pixel_x, pixel_y;     //Touch_getXY() updates global vars
bool Touch_getXY(void)
{
    TSPoint p = ts.getPoint();
    pinMode(YP, OUTPUT);      //restore shared pins
    pinMode(XM, OUTPUT);
    digitalWrite(YP, HIGH);   //because TFT control pins
    digitalWrite(XM, HIGH);
    bool pressed = (p.z > MINPRESSURE && p.z < MAXPRESSURE);
    if (pressed) {
        pixel_x = map(p.x, TS_LEFT, TS_RT, 0, tft.width()); //.kbv makes sense to me
        pixel_y = map(p.y, TS_TOP, TS_BOT, 0, tft.height());
    }
    return pressed;
}
```

Fig 7. Arduino code getting touch points



```
void setup(void)
{
    Serial.begin(9600);
    pinMode(home1,OUTPUT);
      pinMode(home2,OUTPUT);
        pinMode(home3,OUTPUT);
      pinMode(home4,OUTPUT);
    uint16_t ID = tft.readID();
    if (ID == 0xD3D3) ID = 0x9486; // write-only shield
    tft.begin(ID);
    tft.setRotation(0);            //PORTRAIT
    tft.fillScreen(BLACK);//left.up.size.size
    tft.setFont(&FreeMonoBold9pt7b);
    on_btn.initButton(&tft,  105, 90, 190, 40, GREE, BLACK, GREE, "EFY Office 1", 1);
    off_btn.initButton( &tft, 105, 140, 190, 40, AQUA, BLACK, AQUA, "OSFY Office", 1);
    on_btn1.initButton( &tft, 105, 190, 190, 40, WALL, BLACK, WALL, "M.r AK Home", 1);
    off_btn1.initButton(&tft,105, 240, 190, 40, MAROON, BLACK, MAROON, "M.r R.K Home", 1);


    on_btn.drawButton(false);
    off_btn.drawButton(false);
    on_btn1.drawButton(false);
    off_btn1.drawButton(false);
```

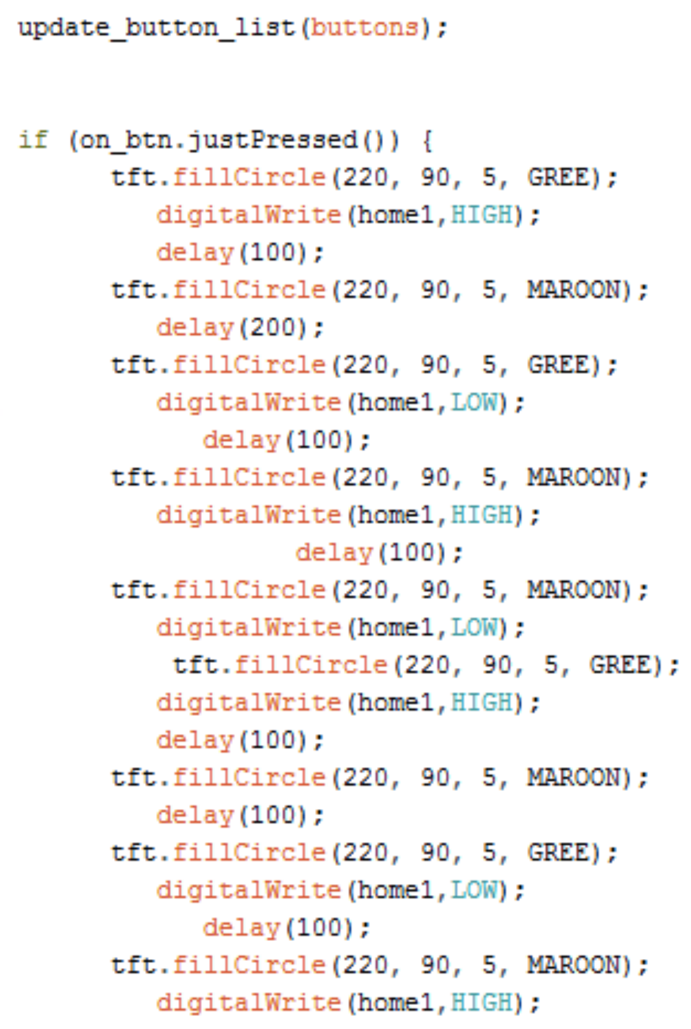
Fig 8. Arduino Code Displaying Buttons



```
Adafruit_GFX_Button *buttons[] = {&on_btn, &off_btn,&on_btn1, &off_btn1, NULL};


bool update_button(Adafruit_GFX_Button *b, bool down)
{
    b->press(down && b->contains(pixel_x, pixel_y));
    if (b->justReleased())
        b->drawButton(false);
    if (b->justPressed())
        b->drawButton(true);
    return down;
}


bool update_button_list(Adafruit_GFX_Button **pb)
{
    bool down = Touch_getXY();
    for (int i = 0 ; pb[i] != NULL; i++) {
        update_button(pb[i], down);
    }
    return down;
}
```

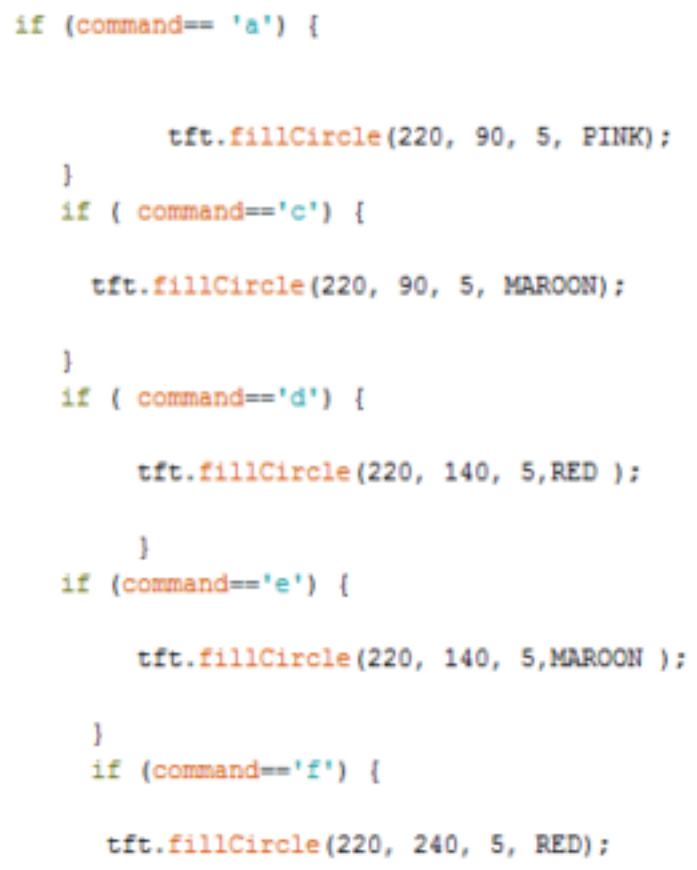
Fig 9. Checking Button status

charger or any 9 volt battery). When you power the Arduino, you will get a menu with several names on your TFT screen. Touch the name where you want to ring the bell. If you want to change the status of the office and house like office is closed or out of home, you can simply connect the Bluetooth to any Bluetooth terminal app available at play store and send the commands as you have set previously in code.



```
update_button_list(buttons);


if (on_btn.justPressed()) {
      tft.fillCircle(220, 90, 5, GREE);
         digitalWrite(home1,HIGH);
         delay(100);
      tft.fillCircle(220, 90, 5, MAROON);
         delay(200);
      tft.fillCircle(220, 90, 5, GREE);
         digitalWrite(home1,LOW);
            delay(100);
      tft.fillCircle(220, 90, 5, MAROON);
         digitalWrite(home1,HIGH);
                  delay(100);
      tft.fillCircle(220, 90, 5, MAROON);
         digitalWrite(home1,LOW);
          tft.fillCircle(220, 90, 5, GREE);
         digitalWrite(home1,HIGH);
         delay(100);
      tft.fillCircle(220, 90, 5, MAROON);
         delay(100);
      tft.fillCircle(220, 90, 5, GREE);
         digitalWrite(home1,LOW);
            delay(100);
      tft.fillCircle(220, 90, 5, MAROON);
         digitalWrite(home1,HIGH);
```

Fig 10. Setting the if condition



```
if (command== 'a') {


          tft.fillCircle(220, 90, 5, PINK);
   }
   if ( command=='c') {

     tft.fillCircle(220, 90, 5, MAROON);

   }
   if ( command=='d') {

        tft.fillCircle(220, 140, 5,RED );

        }
   if (command=='e') {

        tft.fillCircle(220, 140, 5,MAROON );

     }
     if (command=='f') {

      tft.fillCircle(220, 240, 5, RED);
```

Fig 11. Code for setting status

Connection

Arduino	BUZZER
GND	(BUZZER 1,2,3,4) -VE
Pin 13	BUZZER 1 VCC
PIN 12	BUZZER 2 VCC
PIN 11	BUZZER 3 VCC
PIN 13	BUZZER 4 VCC

Arduino	Bluetooth
5V	VCC
GND	GND
RX	TX
TX	RX

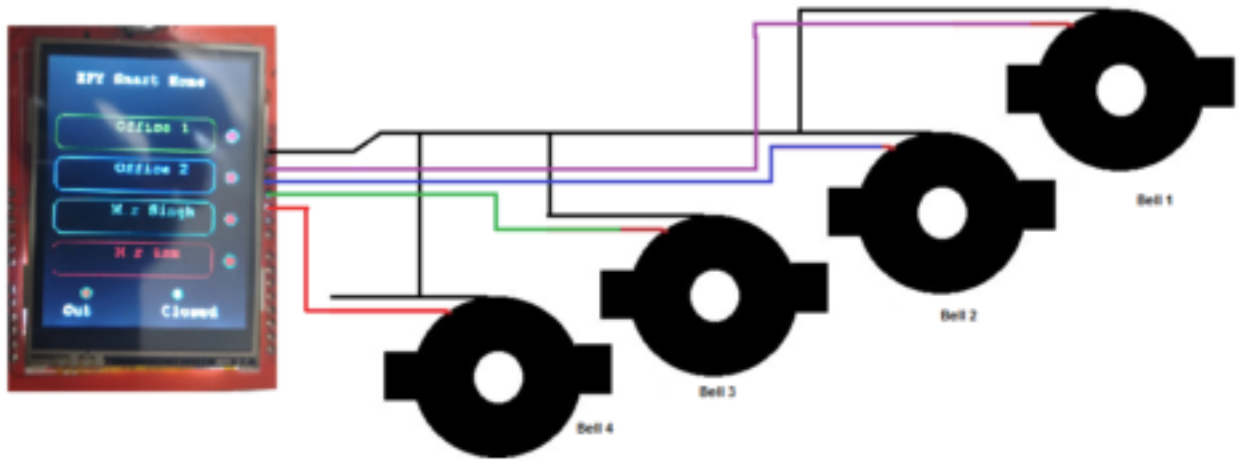


Fig 12.connection

Download Source Folder

You can watch Video for this DIY here: https://youtu.be/IUazM84FDQw

33

SMART BLOOD OXYGEN AND HEART RATE MONITOR WITH AUTOMATIC DATA SAVING SYSTEM

BY ASHWINI KUMAR SINHA

Thanks to the various wearable health devices available in the market today, we can easily monitor our health status at home, without visiting the doctor. How about making one yourself?

In this DIY project, we will try to make a Smart Health Monitoring Device that can measure SpO2 (percentage of oxygen in the blood) and heart rate. This wearable device can be used by athletes to monitor their heart rate and blood oxygen levels during workout. Best part of this project is that you can connect this device to an app that automatically saves all the sensor data to a text file.

So, let's start the project.

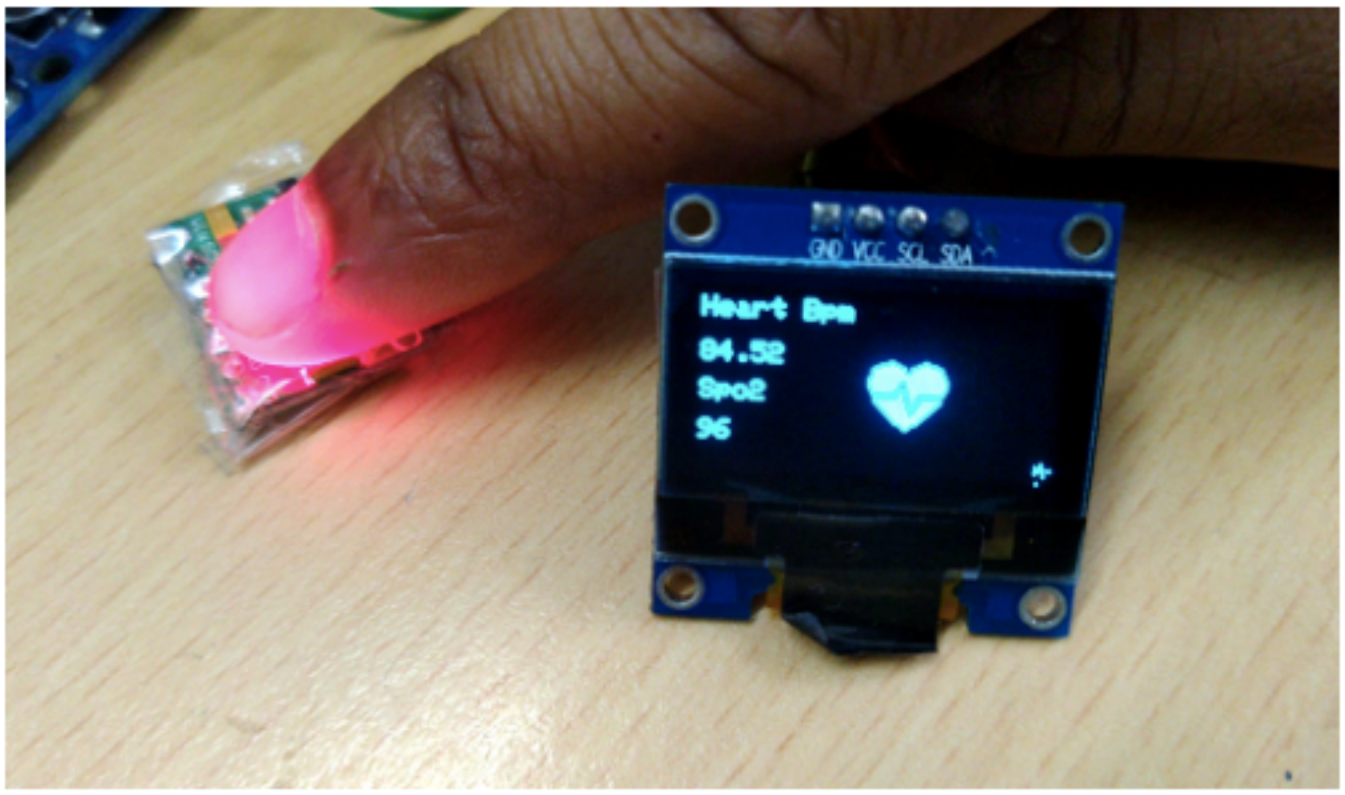


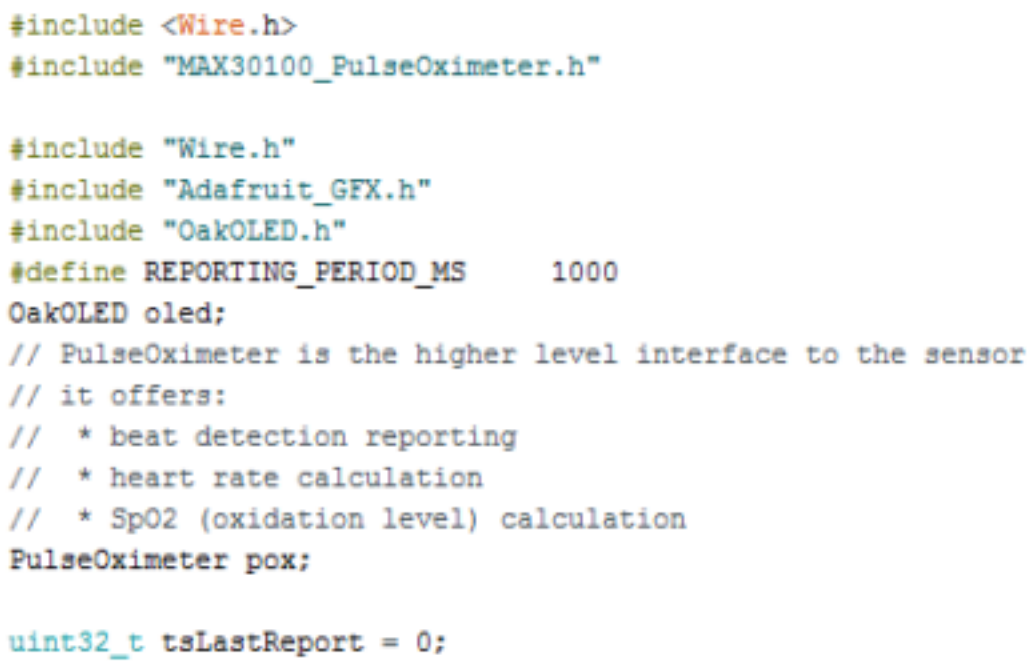
Fig 1. Pulse Sensor

Components:

- Max30100 sensor
- Any Arduino Dev board (Arduino Uno, Nano or pro mini).
- OLED Display (SSD 1306).
- Bluetooth HC 05
- Wires

Coding

First of all, we need to install the required library to the Arduino IDE. Go to tools and open the library manager and search these libraries ("Addafurit GFX", "Oak Oled", "Max 30100") and install these libraries.

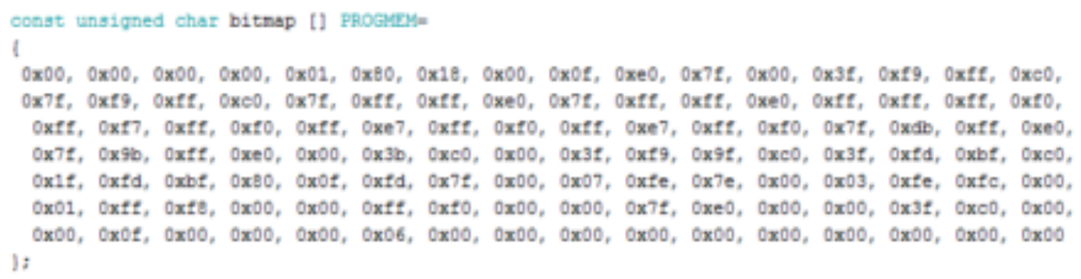


```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"

#include "Wire.h"
#include "Adafruit_GFX.h"
#include "OakOLED.h"
#define REPORTING_PERIOD_MS     1000
OakOLED oled;
// PulseOximeter is the higher level interface to the sensor
// it offers:
//  * beat detection reporting
//  * heart rate calculation
//  * SpO2 (oxidation level) calculation
PulseOximeter pox;

uint32_t tsLastReport = 0;
```

Fig 2. Initiating Libraries in Code

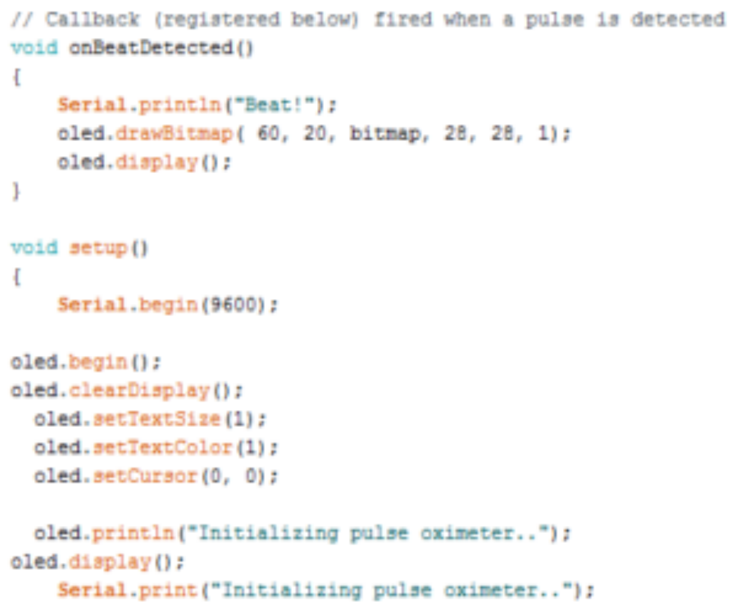
Once we have installed the libraries, we can start coding. In the first part of the code, we have included the required libraries in code.

After this, we will add a short bitmap code that has a heart symbol logo. In the next part, we have a function that displays the heart bitmap logo whenever your heart beats (Refer Fig 3). Now, we can create a setup function and set the baud rate of Bluetooth along with other settings.(Refer Fig 4)



```
const unsigned char bitmap [] PROGMEM=
{
 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x18, 0x00, 0x0f, 0xe0, 0x7f, 0x00, 0x3f, 0xf9, 0xff, 0xc0,
 0x7f, 0xf9, 0xff, 0xc0, 0x7f, 0xff, 0xff, 0xe0, 0x7f, 0xff, 0xff, 0xe0, 0xff, 0xff, 0xff, 0xf0,
  0xff, 0xf7, 0xff, 0xf0, 0xff, 0xe7, 0xff, 0xf0, 0xff, 0xe7, 0xff, 0xf0, 0x7f, 0xdb, 0xff, 0xe0,
  0x7f, 0x9b, 0xff, 0xe0, 0x00, 0x3b, 0xc0, 0x00, 0x3f, 0xf9, 0x9f, 0xc0, 0x3f, 0xfd, 0xbf, 0xc0,
  0x1f, 0xfd, 0xbf, 0x80, 0x0f, 0xfd, 0x7f, 0x00, 0x07, 0xfe, 0x7e, 0x00, 0x03, 0xfe, 0xfc, 0x00,
  0x01, 0xff, 0xf8, 0x00, 0x00, 0xff, 0xf0, 0x00, 0x00, 0x7f, 0xe0, 0x00, 0x00, 0x3f, 0xc0, 0x00,
  0x00, 0x0f, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

Fig 3. Bitmap Code



```
// Callback (registered below) fired when a pulse is detected
void onBeatDetected()
{
    Serial.println("Beat!");
    oled.drawBitmap( 60, 20, bitmap, 28, 28, 1);
    oled.display();
}

void setup()
{
    Serial.begin(9600);

oled.begin();
oled.clearDisplay();
  oled.setTextSize(1);
  oled.setTextColor(1);
  oled.setCursor(0, 0);

  oled.println("Initializing pulse oximeter..");
oled.display();
    Serial.print("Initializing pulse oximeter..");
```

Fig 4. Arduino code setting baud rate of Bluetooth

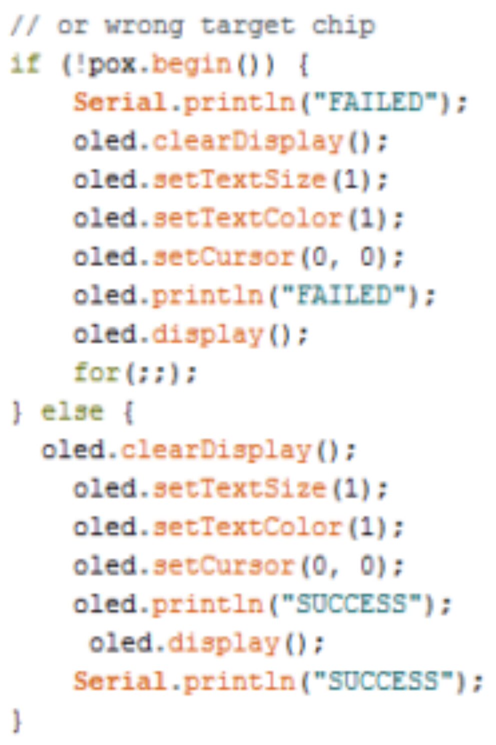
Then we will create a loop function that updates the readings of sensor and displays those readings on OLED screen.

Now, connect the components as illustrated at the end of the page.

App Making

We are done with coding and connection parts, so let's build the app. We are going to use MIT App Inventor. First, we will create a layout and add the following components to it.

- An Image View
- A Button
- A Text Level
- A Text input
- Bluetooth Clint
- Clock Timmer



```
// or wrong target chip
if (!pox.begin()) {
    Serial.println("FAILED");
    oled.clearDisplay();
    oled.setTextSize(1);
    oled.setTextColor(1);
    oled.setCursor(0, 0);
    oled.println("FAILED");
    oled.display();
    for(;;);
} else {
  oled.clearDisplay();
    oled.setTextSize(1);
    oled.setTextColor(1);
    oled.setCursor(0, 0);
    oled.println("SUCCESS");
     oled.display();
    Serial.println("SUCCESS");
}
```

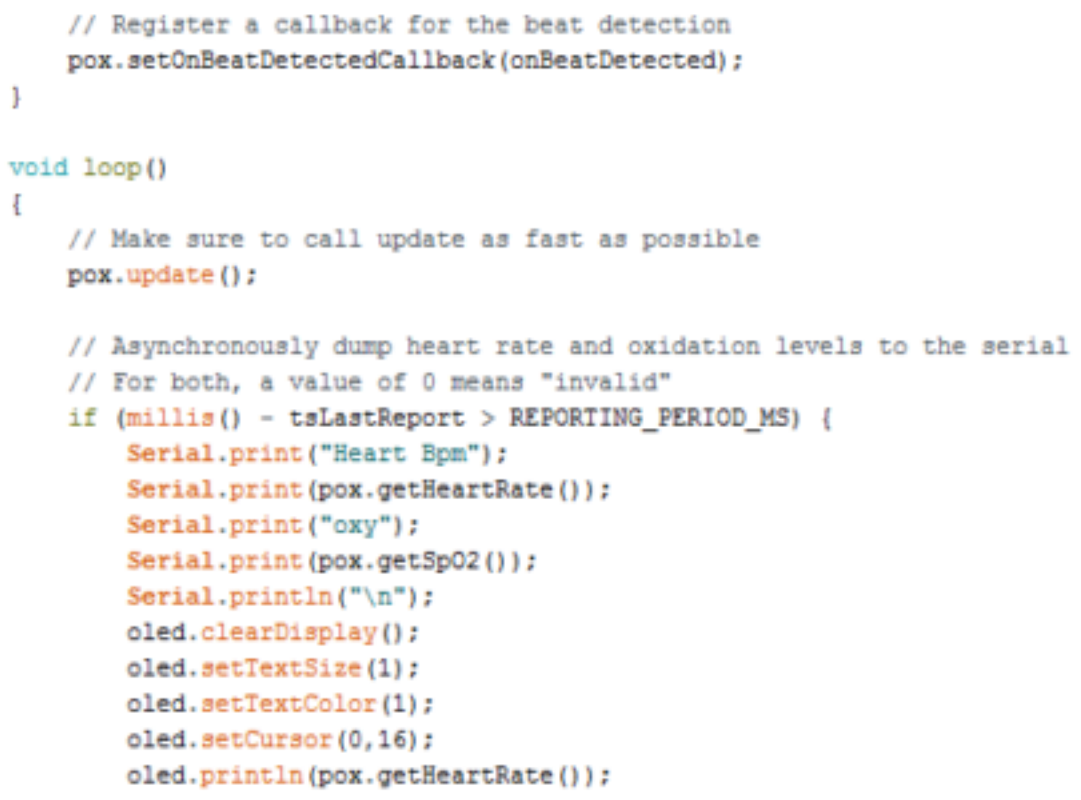
Fig 5. Code for showing sensor status

• File Storage

Then Join the MIT App inventor code blocks according to Fig 9.

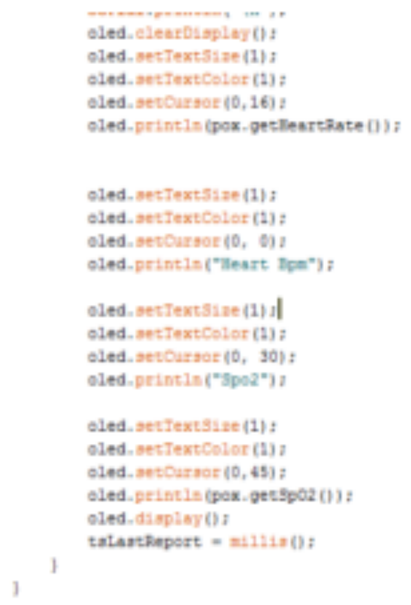
Testing

After connecting the components, recheck each connection. Then power the Arduino. When you power the Arduino, you get a message on OLED screen which says - “Initializing Sensor.” After a few seconds when Arduino detects the sensor, it shows the reading on the screen. Initially, it will show 0 or false reading. Now place your finger on the sensor and let it detect your heartbeat rate and blood oxygen level. After a few seconds, your SpO2 (oxygen percentage in the blood) and heartbeat rate will be displayed on the OLED screen. Next, open the app that you have made and connect it to the



```
    // Register a callback for the beat detection
    pox.setOnBeatDetectedCallback(onBeatDetected);
}

void loop()
{
    // Make sure to call update as fast as possible
    pox.update();

    // Asynchronously dump heart rate and oxidation levels to the serial
    // For both, a value of 0 means "invalid"
    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {
        Serial.print("Heart Bpm");
        Serial.print(pox.getHeartRate());
        Serial.print("oxy");
        Serial.print(pox.getSpO2());
        Serial.println("\n");
        oled.clearDisplay();
        oled.setTextSize(1);
        oled.setTextColor(1);
        oled.setCursor(0,16);
        oled.println(pox.getHeartRate());
```

Fig 6. Arduino code for getting sensor data



```
        oled.clearDisplay();
        oled.setTextSize(1);
        oled.setTextColor(1);
        oled.setCursor(0,16);
        oled.println(pox.getHeartRate());


        oled.setTextSize(1);
        oled.setTextColor(1);
        oled.setCursor(0, 0);
        oled.println("Heart Bpm");

        oled.setTextSize(1);
        oled.setTextColor(1);
        oled.setCursor(0, 30);
        oled.println("Spo2");

        oled.setTextSize(1);
        oled.setTextColor(1);
        oled.setCursor(0,45);
        oled.println(pox.getSpO2());
        oled.display();
        tsLastReport = millis();
    }
}
```

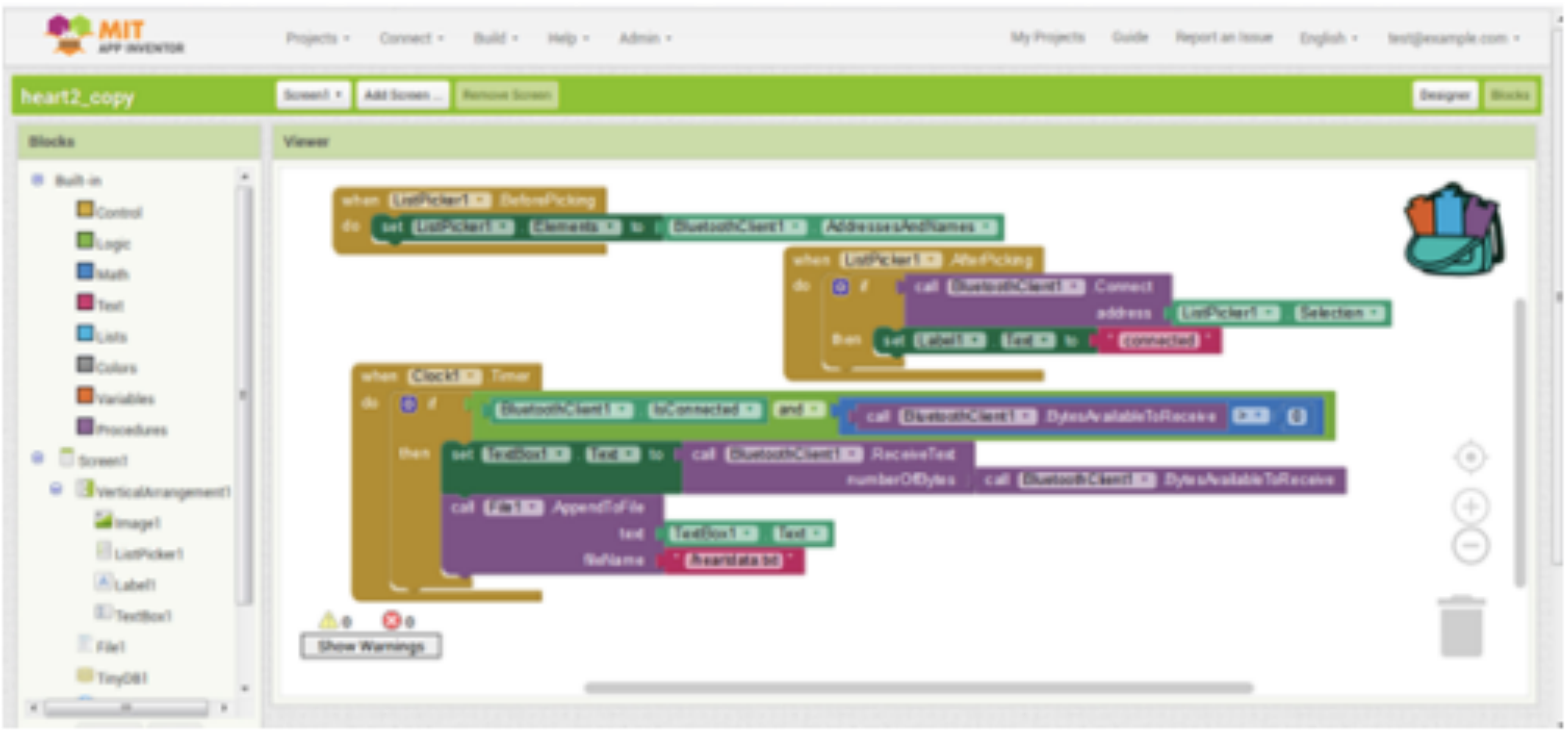
Fig 7. Arduino code for displaying Sensor data on OLED

Arduino	Components
5v	OLED and Max30100 VCC
GND	OLED and Max30100 GND
SCL	SCL of OLED and Max 30100
SDA	SDA of OLED and Max30100
5V	Bluetooth VCC
GND	Bluetooth GND
RX	TX
TX	RX

Bluetooth. After it is connected, you can see the blood oxygen percentage (SpO2) and pulse rate in the app. The app automatically saves the sensor data in the form of text file, which you can share with a doctor or use it for analysis.

Connection

Arduino	Components
5v	OLED and Max30100 VCC
GND	OLED and Max30100 GND
SCL	SCL of OLED and Max 30100
SDA	SDA of OLED and Max30100
5V	Bluetooth VCC
GND	Bluetooth GND
RX	TX
TX	RX

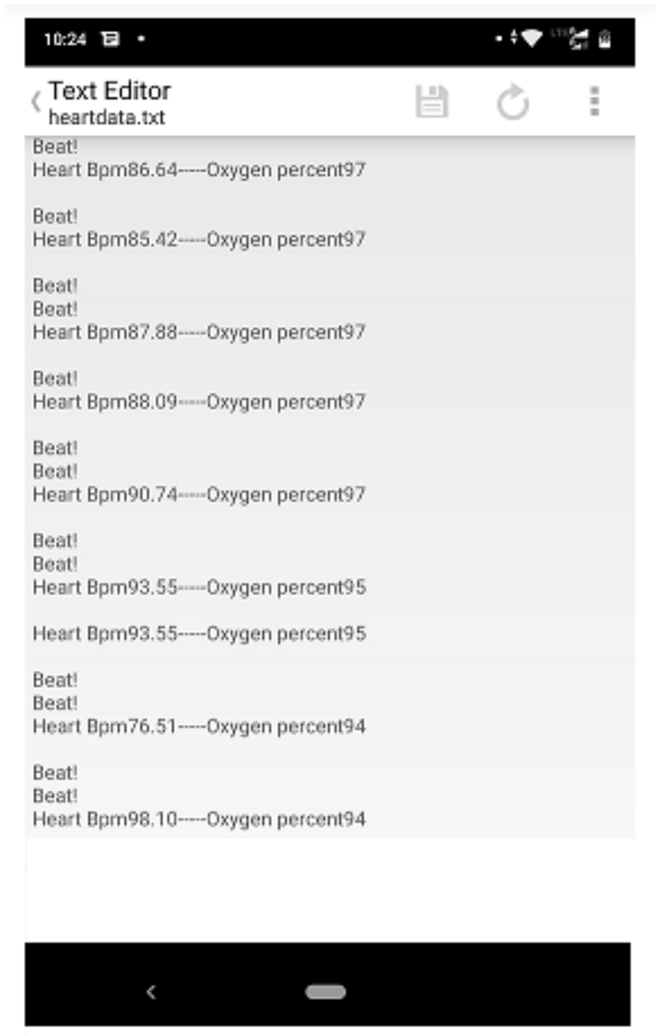


```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  if  call BluetoothClient1 .Connect
                               address  ListPicker1 . Selection
    then  set Label1 . Text to " connected "

when Clock1 .Timer
do  if  BluetoothClient1 . IsConnected  and  call BluetoothClient1 .BytesAvailableToReceive  >  0
    then  set TextBox1 . Text to  call BluetoothClient1 .ReceiveText
                                                  numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive
          call File1 .AppendToFile
                          text  TextBox1 . Text
                      fileName  " heartdata.txt "
```

Fig 9. App Inventor code Blocks

Download Source Code

You can watch Video for this DIY here: https://youtu.be/47oL-MT51LQ



```
Beat!
Heart Bpm86.64-----Oxygen percent97

Beat!
Heart Bpm85.42-----Oxygen percent97

Beat!
Beat!
Heart Bpm87.88-----Oxygen percent97

Beat!
Heart Bpm88.09-----Oxygen percent97

Beat!
Beat!
Heart Bpm90.74-----Oxygen percent97

Beat!
Beat!
Heart Bpm93.55-----Oxygen percent95

Heart Bpm93.55-----Oxygen percent95

Beat!
Beat!
Heart Bpm76.51-----Oxygen percent94

Beat!
Beat!
Heart Bpm98.10-----Oxygen percent94
```

Fig 10. Sensor data in text

34

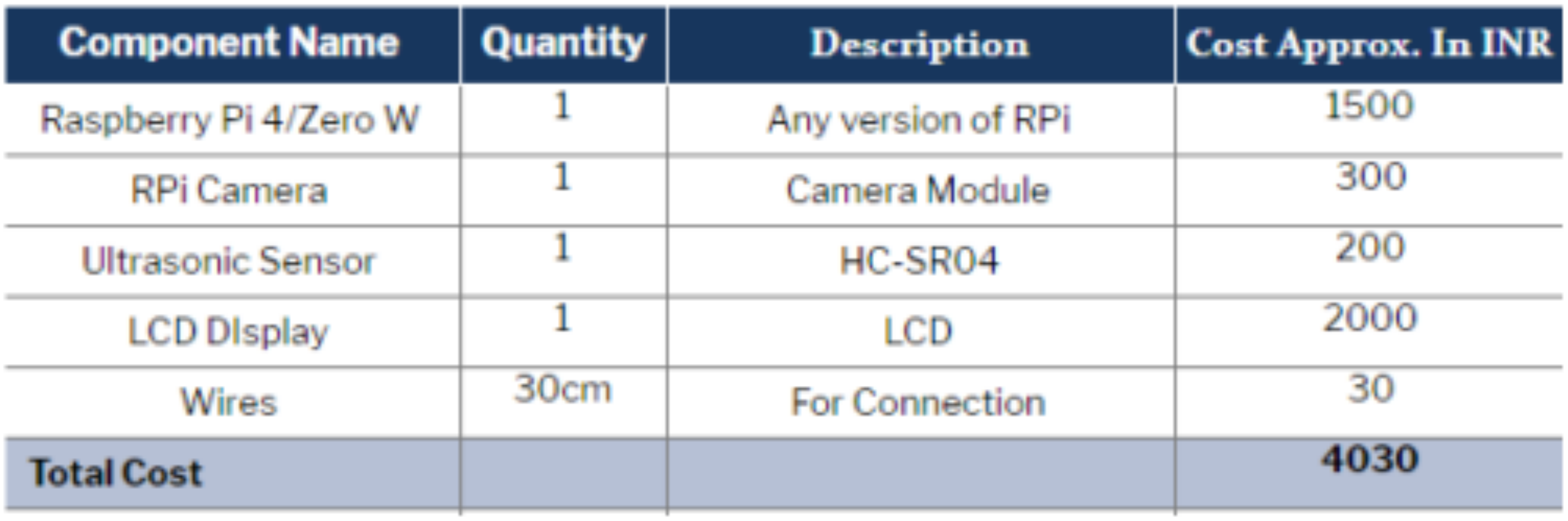
SMART CAR BACK CAMERA AND COLLISION PROTECTION

BY ASHWINI KUMAR SINHA

Most people use the side and rear-view mirrors to manoeuvre their cars since a camera installed at the back of the car can be quite expensive. So, today we have undertaken the task of creating a smart back-camera system for cars. Our system can also be used as a smart car dashboard display that can play music, video and do a lot of things that a computer can do.

In this project, we will use Raspberry pi to connect the camera and display to the video of the back of the car. We will also add an ultrasonic sensor that measures the distance from our car to any obstacle behind the car. So let's start our project by shopping of following components.

Bill of Materials



Component Name	Quantity	Description	Cost Approx. In INR
Raspberry Pi 4/Zero W	1	Any version of RPi	1500
RPi Camera	1	Camera Module	300
Ultrasonic Sensor	1	HC-SR04	200
LCD DIsplay	1	LCD	2000
Wires	30cm	For Connection	30
Total Cost			**4030**

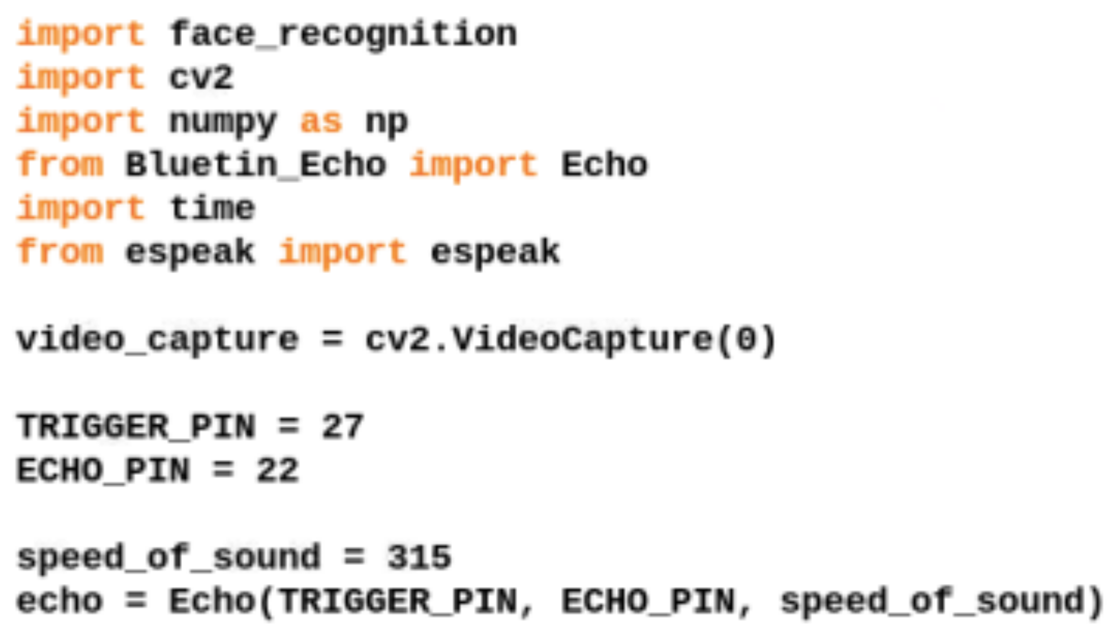
Coding

Before starting with the code, we need to install the libraries and python modules. To do this, open the raspbian terminal and use the following commands to install the python modules

```
sudo  pip3 install  opencv
sudo pip3 install Bluetin_Echo
sudo pip3 install espeak
```

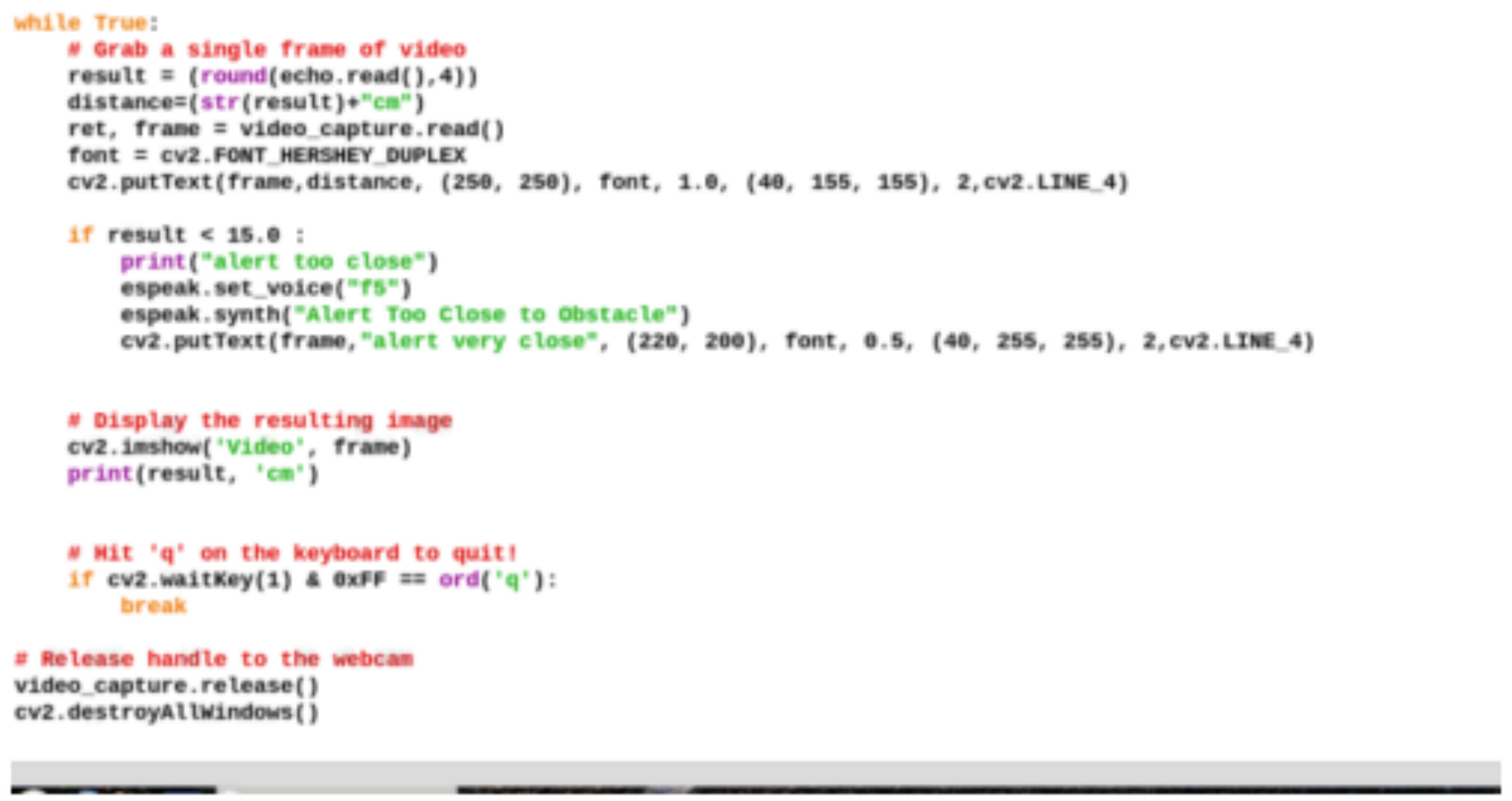
Now, after installation of modules, open the python3 IDLE and create the following code

First import the required libraries (espeak, cv2, time, Bluetin_Echo) in code, then set the ‘echo’ and ‘trigger’ pin to ‘no’. For the ultrasonic sensor, we’ve used the pin 27 for ‘echo’ and 22 for ‘trigger’. Next, we set the sound speed and then start the ‘echo’ and ‘trigger’ signal at the speed of sound. After that, we need to create a ‘while’ loop that will run repeatedly. In this loop, we will check the distance between the car and the obstacle using the ultrasonic sensor reading. Next, we try to display the video frame camera with distance reading on the screen. After that, we have created an 'if' condition that checks whether the distance between the car and the obstacle is under 15cm or not (this threshold can be changed). If it returns true, the LCD display will alert us, as will the speaker. (Refer Fig 2,3)



```
import face_recognition
import cv2
import numpy as np
from Bluetin_Echo import Echo
import time
from espeak import espeak

video_capture = cv2.VideoCapture(0)

TRIGGER_PIN = 27
ECHO_PIN = 22

speed_of_sound = 315
echo = Echo(TRIGGER_PIN, ECHO_PIN, speed_of_sound)
```

Fig 2



```
while True:
    # Grab a single frame of video
    result = (round(echo.read(),4))
    distance=(str(result)+"cm")
    ret, frame = video_capture.read()
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(frame,distance, (250, 250), font, 1.0, (40, 155, 155), 2,cv2.LINE_4)

    if result < 15.0 :
        print("alert too close")
        espeak.set_voice("f5")
        espeak.synth("Alert Too Close to Obstacle")
        cv2.putText(frame,"alert very close", (220, 200), font, 0.5, (40, 255, 255), 2,cv2.LINE_4)


    # Display the resulting image
    cv2.imshow('Video', frame)
    print(result, 'cm')


    # Hit 'q' on the keyboard to quit!
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release handle to the webcam
video_capture.release()
cv2.destroyAllWindows()
```

Fig 3.

Connection

Now connect the ultrasonic sensor as in the circuit diagram (Refer Fig 4). Then the camera in with a raspberry pi using the ribbon cable after this you set the camera and ultrasonic sensor in the back part of the car.

Next, attach the car aux input to RPI audio output jack. Then connect the raspberry pi display to its HDMI input if your display is HDMI based or RCA input if your display is RCA cable based.

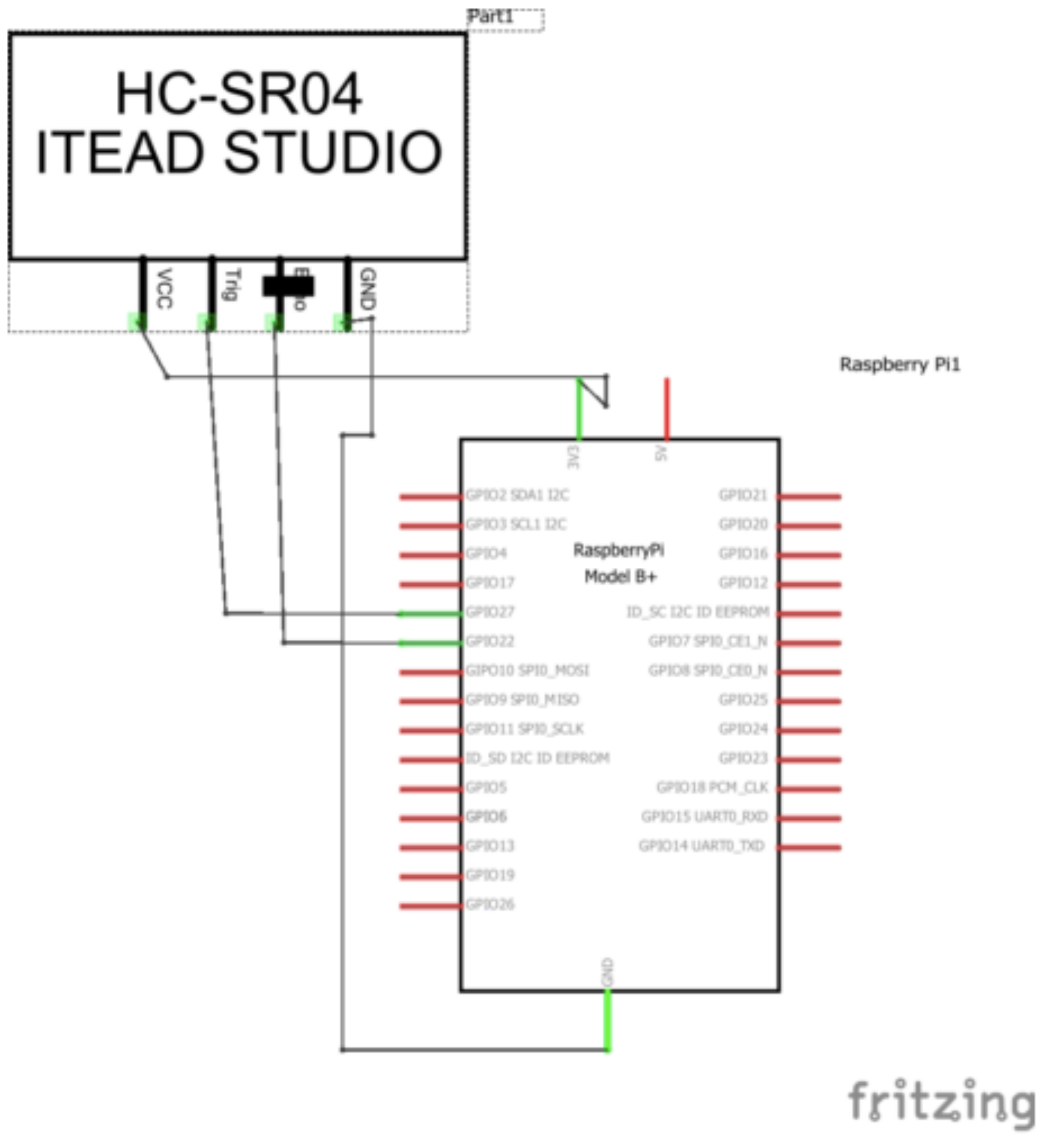
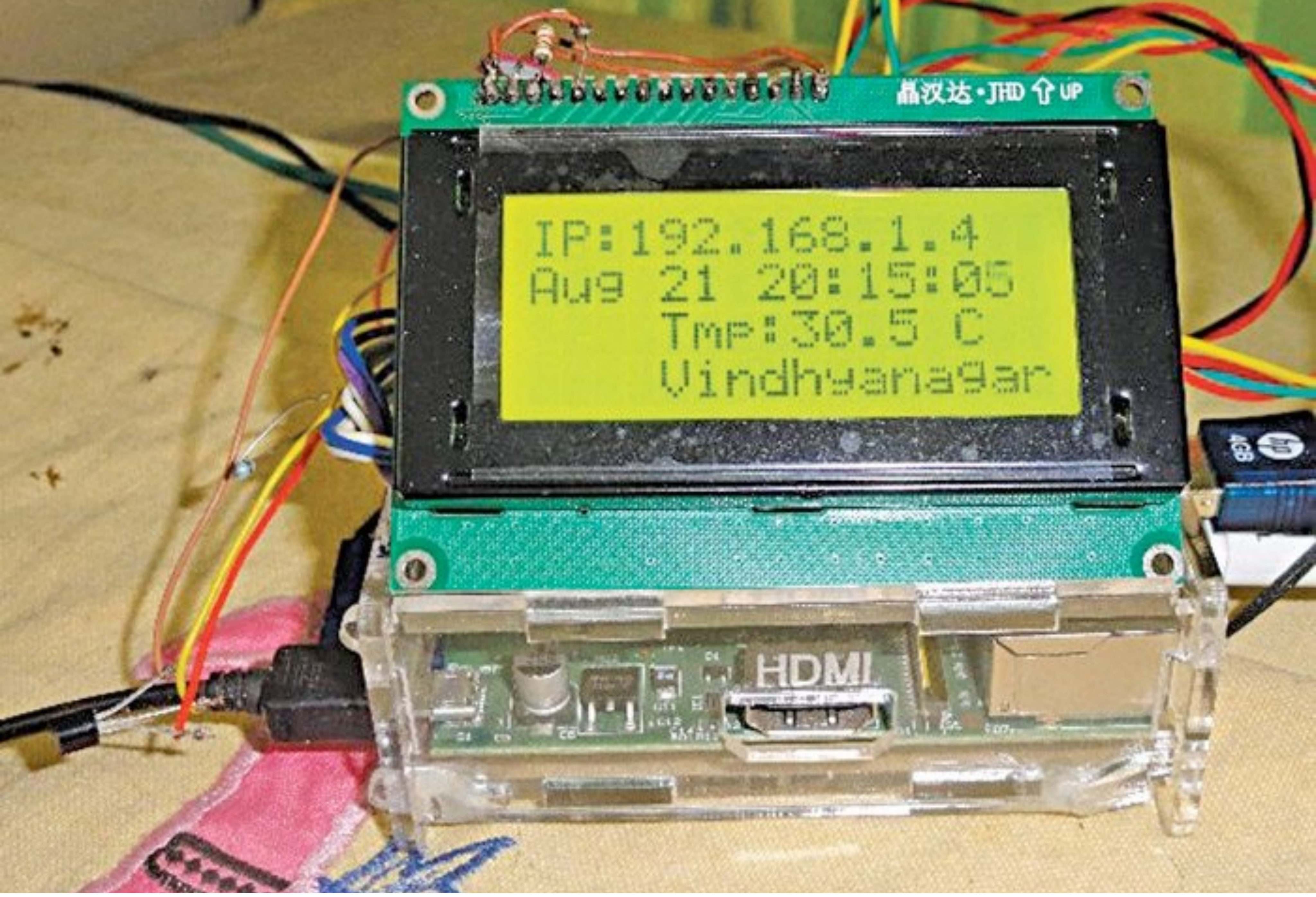


Fig 4.Connection

Testing

Power the PI with 5V and then run the code we have created. After a few seconds, you will see the window display the camera video with distance readings. Now when you come too close, the system will start alerting through speakers saying "Too close to collision " and same will be displayed on the camera video.

Download Source Folder



35

MULTIFUNCTION LCD CLOCK WITH RASPBERRY PI

BY BY SOMNATH BERA

With just two simple Python files, an inexpensive LCD display and a low-cost digital temperature sensor, you can turn Raspberry Pi into a beautiful LCD clock with IP address and temperature indication facility. There will still be a few GPIOs available to hook up other systems such as an alarm triggered by temperature or time threshold. Since we will be using quite a handful of data, it is better to use a four-line LCD display with characters not less than 16 per line. Fig. 1 shows the author's prototype.

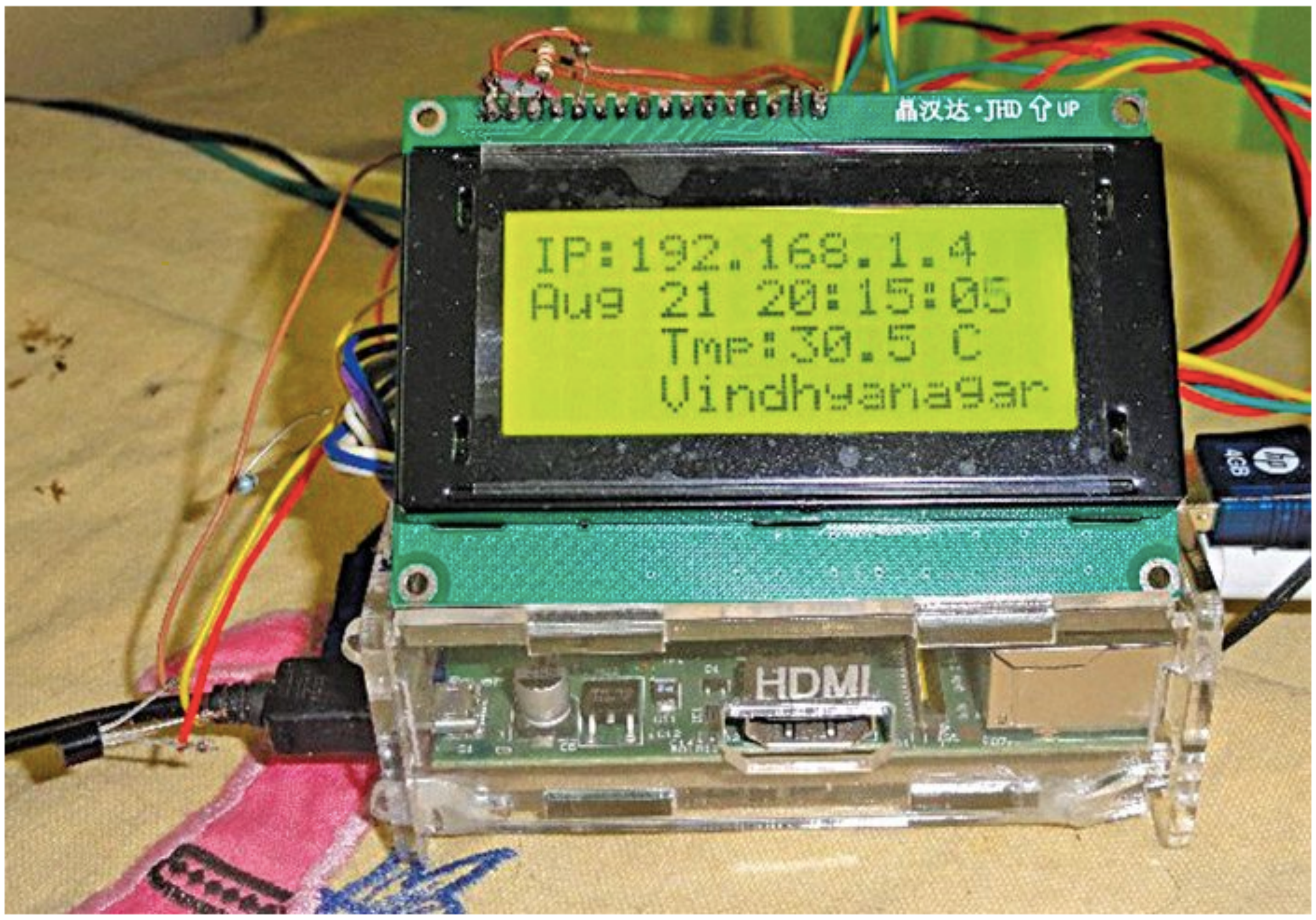


Fig. 1: Author's prototype for Multifunction LCD Clock

Circuit and working

Fig. 2 shows the pin configuration of the Raspberry Pi. The GPIO connector has seven true GPIO pins, I2C interface, SPI interface, serial TX/RX pins and PWM pins that can be used to control external hardware.

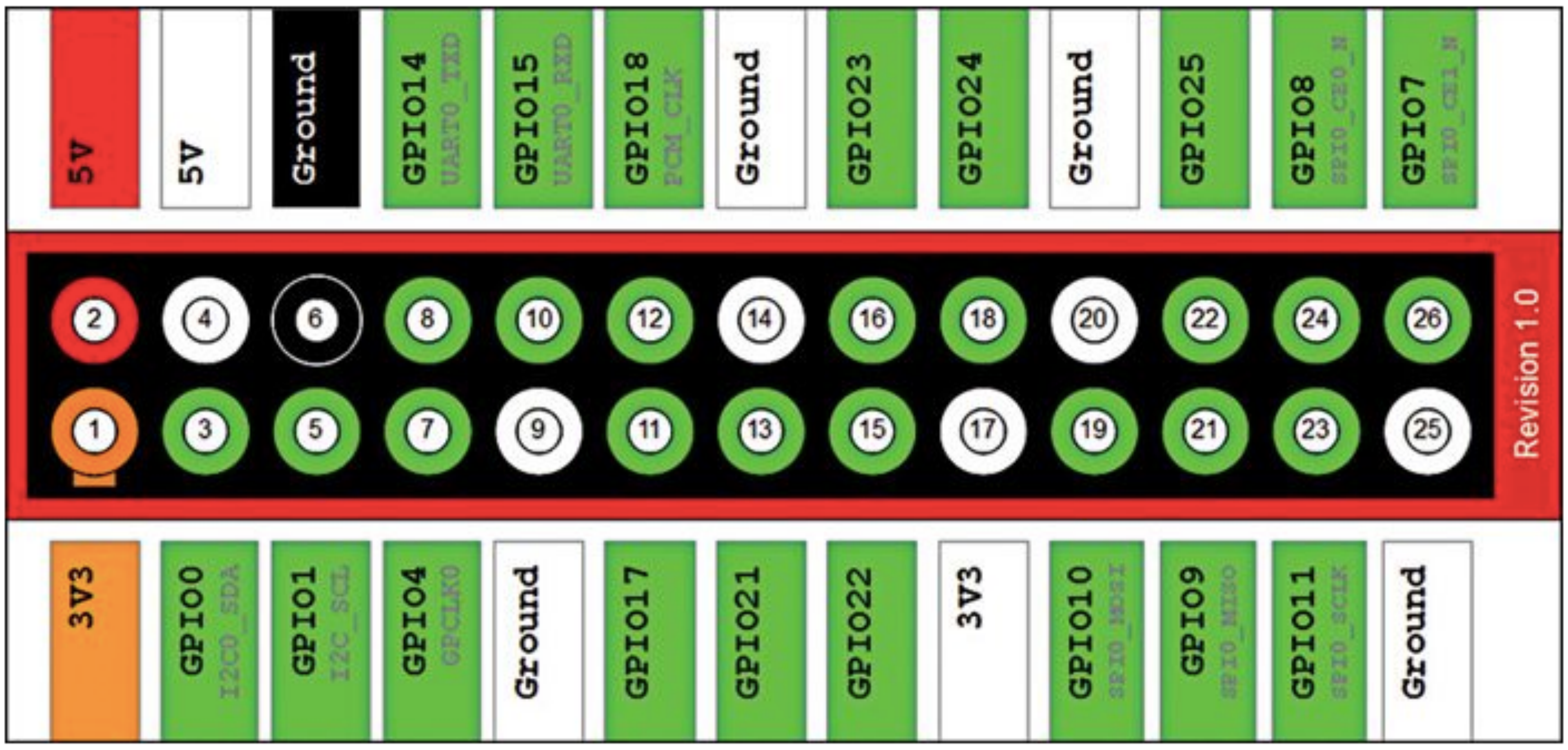


Fig. 2: Pin configuration of Raspberry Pi

Fig. 3 shows the connection scheme for the clock. The main components used are Raspberry Pi board, temperature sensor DS18B20 (IC1) and 4×20 LCD module. IC1 is a heat sensor with an operating range of -55°C to +125°C. It can derive power directly from the data line, eliminating the need for an external power supply. But, in this project, we have used 3.3V output from Raspberry Pi to power it. The system works on one-wire protocol. Output of temperature sensor IC1 is read by GPIO4 (pin 7) of Raspberry Pi.

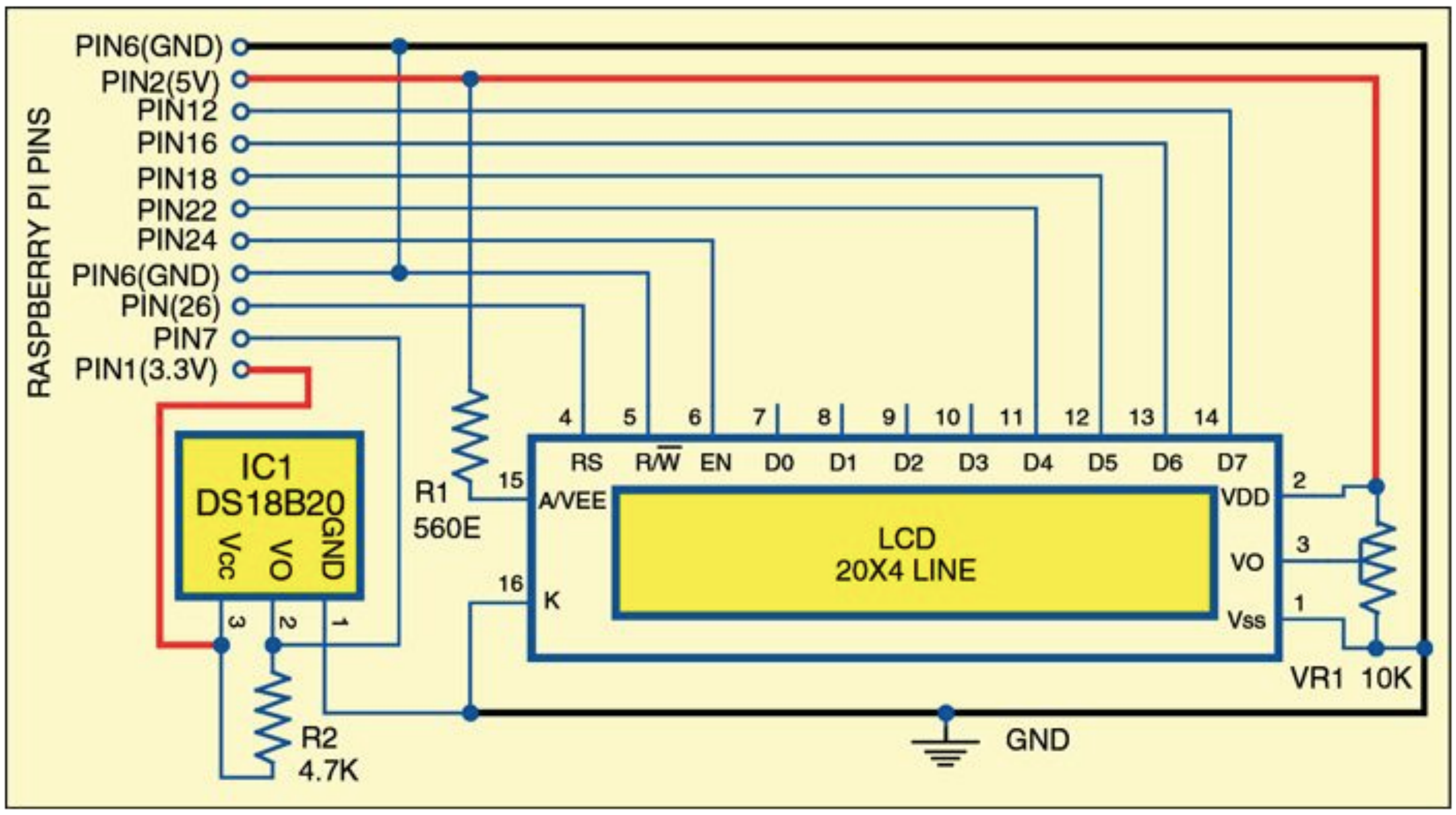


Fig. 3: Connection scheme for multifunction LCD clock with Raspberry Pi

Make all the connections as per Fig. 3 and ensure that a small resistor (4.7K to 10K) is connected between data pin 2 and power pin 3 of IC1 to pull-down data-out signal.

Switch on your Raspberry Pi and touch the sensor (IC1) to see that it is not getting over-heated because of any wrong connection. Once all the connections are checked, follow the instructions in software section below to load the temperature sensor module and the clock. The IP and temperature will appear on the LCD.

Software

It is assumed that your Raspberry Pi is already set up with Raspbian 'wheezy' operating system. If not, you can refer 'Getting Started with Raspberry Pi' article published in April 2013 issue to set it up.

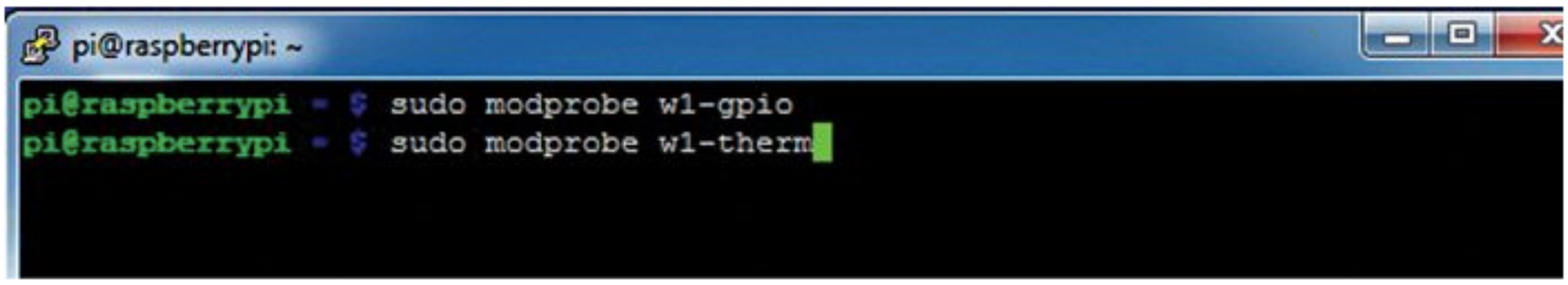
Now, all you need is a network connection for Raspberry Pi to install all the software. Refer 'Set Up Network for Raspberry Pi' published in May 2013 issue for getting the network connection up on your Raspberry Pi. Once done, you can either connect a keyboard and a display to Raspberry Pi and start following the installations using Lx terminal, or you can access Raspberry Pi remotely using SSH and execute all the commands directly.

After login to Raspberry Pi, load the DS18B20 module and get the temperature data ready for viewing, using the commands:

$ sudo modprobe w1-gpio
$ sudo modprobe w1-therm

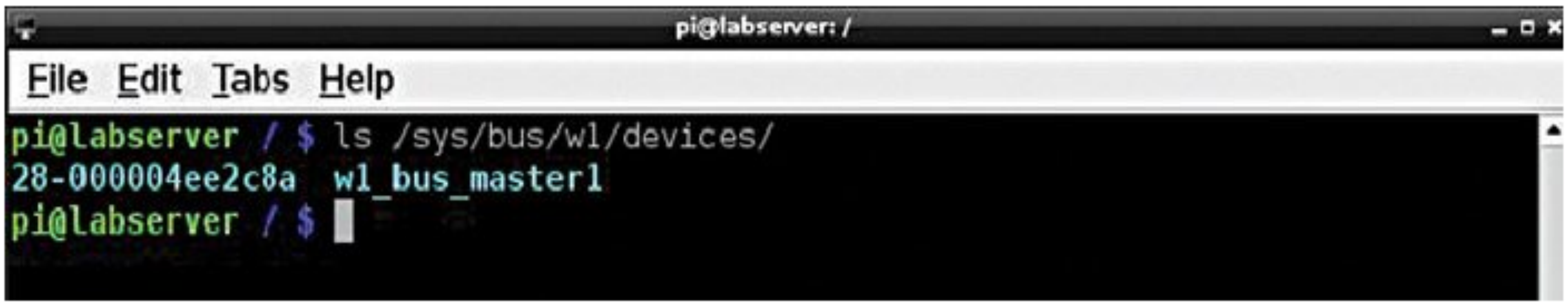
You will need ID number of the DS18B20 module to read the temperature through the application program. You can get the ID using the command mentioned below, as shown in Fig. 5:

$ ls /sys/bus/w1/devices/



```
pi@raspberrypi ~ $ sudo modprobe w1-gpio
pi@raspberrypi ~ $ sudo modprobe w1-therm
```

Fig. 4: Loading the DS18B20 module



```
pi@labserver / $ ls /sys/bus/w1/devices/
28-000004ee2c8a  w1_bus_master1
pi@labserver / $
```

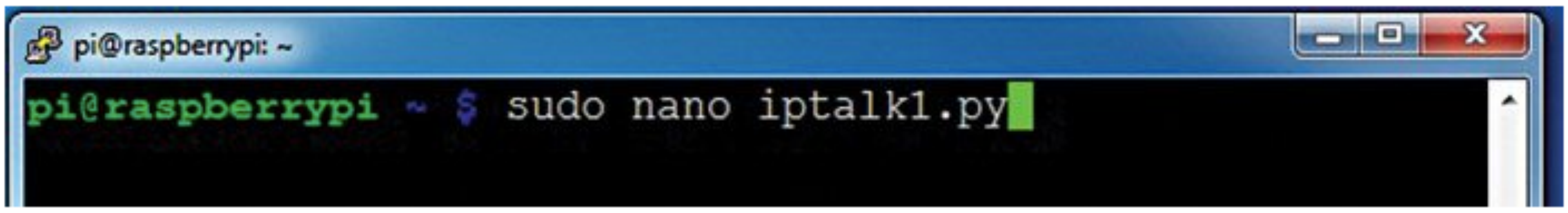
Fig. 5: ID of temperature sensor module

The ID in our case is 28-000004ee2c8a. In your case, it will be similar but certainly different. Note this ID as it will be used in the application program to get the temperature data.

Download Source Code: Click Here

The application program is written using Python programming language. Download the source code (iptalk1.py and iptalk2.py).

Create a new file with the name iptalk1.py and open it in nano editor, using the command mentioned below, as shown in Fig. 6:



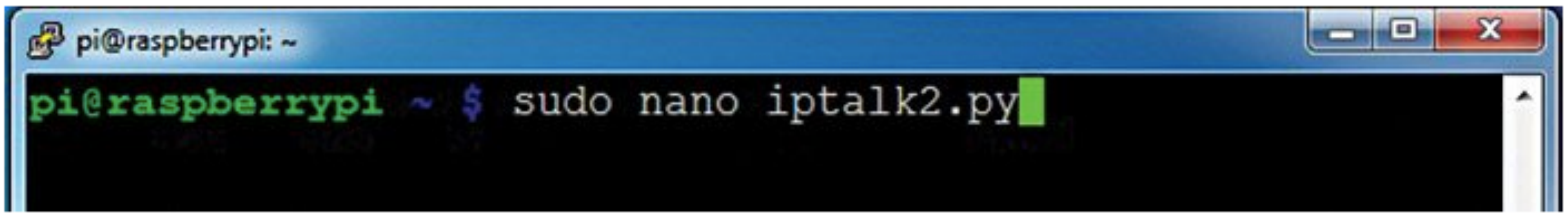
```
pi@raspberrypi ~ $ sudo nano iptalk1.py
```

Fig. 6: Creating file iptalk1.py

$ sudo nano iptalk1.py

Copy the source code from the downloaded file iptalk1.py to this new file created in Raspberry Pi. Save it by using ctrl+o and then exit through ctrl+x.

Similarly, create another file with the name iptalk2.py in the same directory and open it in nano editor, using the command mentioned below, as shown in Fig. 7:



```
pi@raspberrypi ~ $ sudo nano iptalk2.py
```

Fig. 7: Creating file iptalk2.py

$ sudo nano iptalk2.py

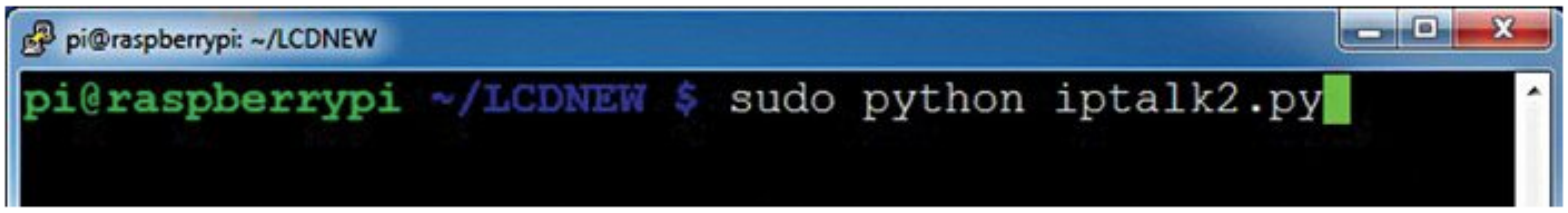
Copy the source code from the downloaded iptalk2.py file to this new file created in Raspberry Pi. In this code, you will have to make the changes for the temperature module ID noted in Fig. 5. Change the ID in the section mentioned below and save the file using ctrl+o and then exit through ctrl+x.

try:

while True:
tfile = open("/sys/bus/w1/
devices/28-0000049582dd/w1_slave")
text = tfile.read()
tfile.close()
secondline = text.split("\n")[1]
temperaturedata = secondline.
split(" ")[9]
temperature =
float(temperaturedata[2:])
temperature = temperature / 1000
temp = 'Tmp:'+str(temperature)+' C'

Please note, the indentation of the copied code in above two files should exactly be the same. Python is sensitive to indentation.

Finally, run the source code using the command below, as shown in Fig. 8, and you will see the LCD display shown in Fig. 1:

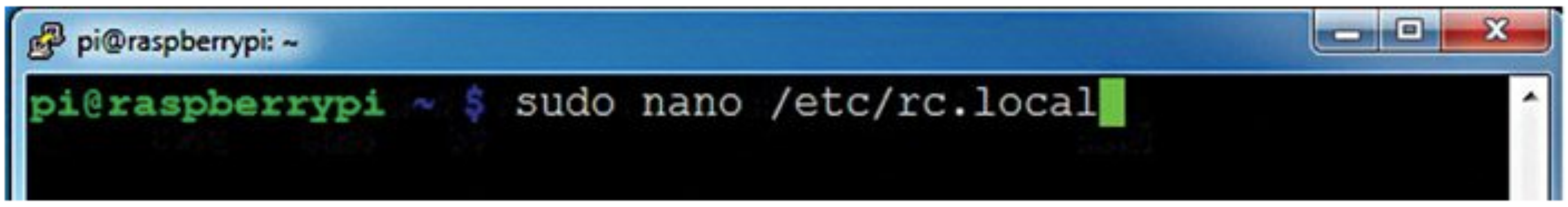


```
pi@raspberrypi: ~/LCDNEW
pi@raspberrypi ~/LCDNEW $ sudo python iptalk2.py
```

Fig. 8: Running the Python code

$ sudo python iptalk2.py

For making it work on every boot, open the '/etc/rc.local' file in nano editor, as shown in Fig. 9. Then, add the line 'sudo python /where-your-file-is/iptalk2.py' in the beginning.



```
pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo nano /etc/rc.local
```

Fig. 9: Editing the rc.local file

Now, every time you boot your Raspberry Pi computer, the program will execute automatically and the LCD will display all the information, as shown in Fig. 1. Touch the temperature sensor and see the temperature on LCD increasing.

36

ARDUINO BASED LIQUID VENDING MACHINE

BY SAGAR RAJ AND MAHENDRA RAISINGHANI

This is an automatic liquid vending machine based on radio-frequency identification (RFID) system. A fixed amount of liquid can be dispensed by swiping an RFID tag across the RFID reader. An alphanumeric LCD is used to display the operation and instructions for users to follow while dispensing the liquid. This machine could be implemented in organisations like hospitals and colleges (medical, engineering, etc) to provide 24-hour service to customers with no humans involved.

The vending machine has many advantages and is highly beneficial in many ways. Some examples are given below:

1. In hospitals, it can be used as a milk vending machine. Patients can buy milk from the vending machine without going to the market.
2. In educational organisations, some shops are opened for fixed hours. With this machine, you get 24-hour service.
3. At events/functions, this machine can be used to dispense juice of varied flavours.
4. In rural areas, this machine can be used to dispense water for a variety of purposes including drinking and watering plants to enhance productivity as well as quality.

The concept of this project could be useful for hobbyists and designers to further design a rugged-mechanical machine. This DIY article describes programming and interfacing the circuit as per the working prototype without mechanical construction parts. The main components required in this project are listed in the table on next page. The block diagram of Arduino based liquid vending machine is shown in Fig. 1.

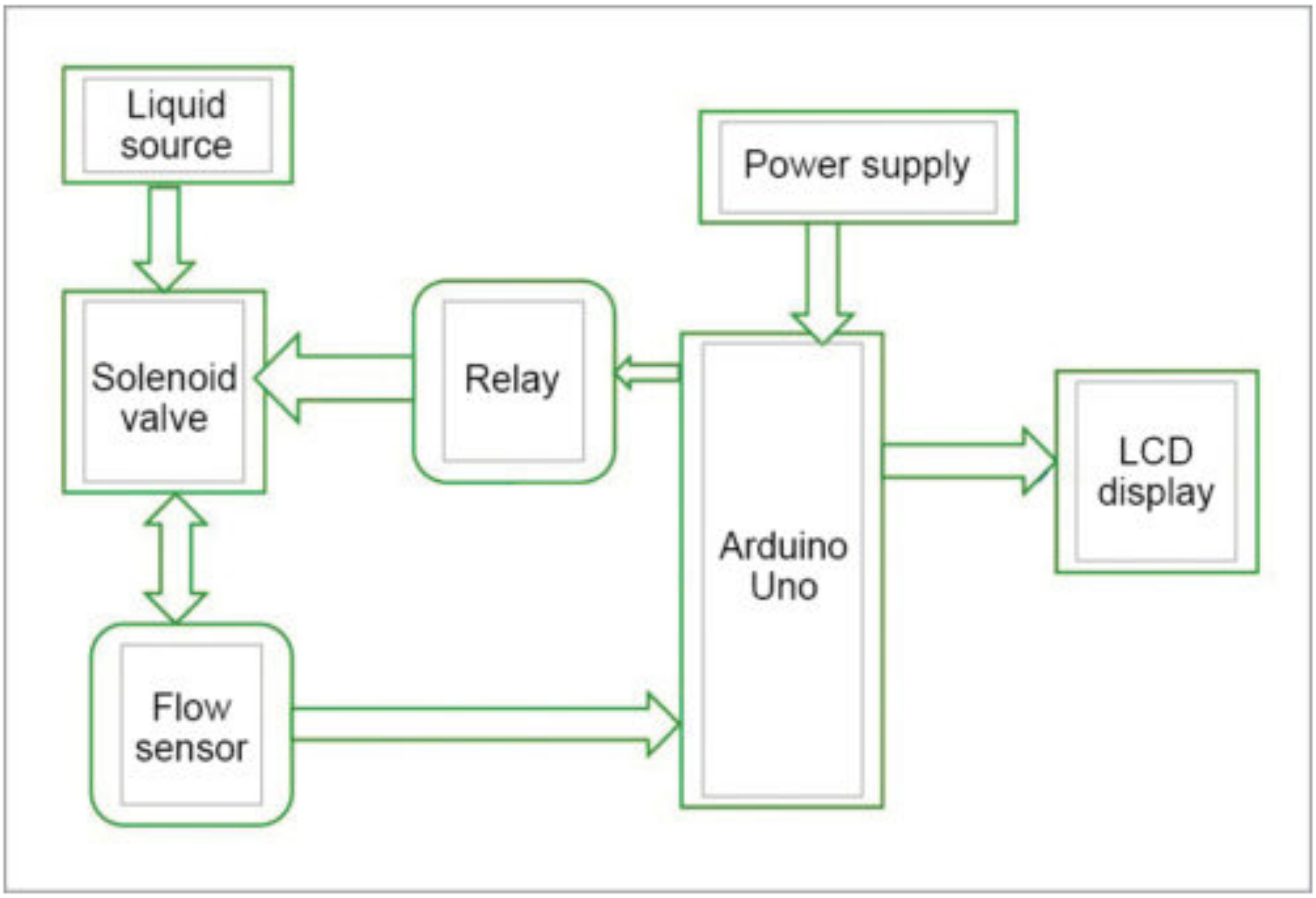


Fig. 1: Block diagram of Arduino based liquid vending machine

Circuit and working

The circuit diagram of Arduino based liquid vending machine is shown in Fig. 2. This project consists of analogue flow sensor, RFID tag, RFID reader, 16x2 LCD, solenoid, Arduino Uno, a single channel relay and a 12V DC power supply.

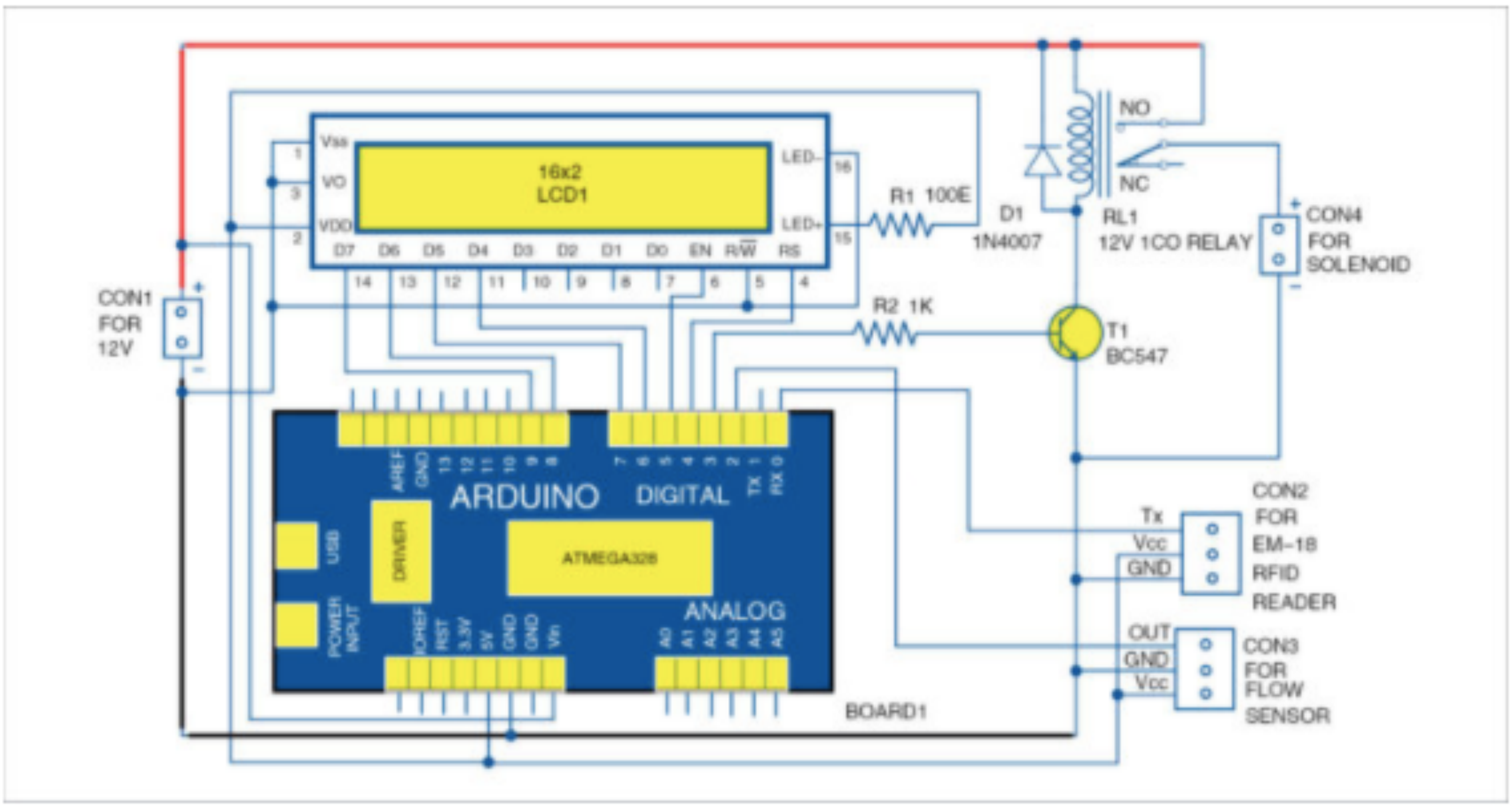


Fig. 2: Circuit diagram of the liquid vending machine

Analogue water flow sensor

This sensor measures the flow rate of a fluid. It has two openings—one opening for fluid in-take and the other for out-take.

It works on the principle of the Hall effect. The Hall effect is utilised in the flow meter using a small fan/propeller-shaped rotor, which is placed in the path of the liquid flow.

It has three wires: red wire for supply voltage, black wire for ground and a yellow wire to collect output from Hall effect sensor. Supply voltage could be from 5V to 18V DC.

RFID reader

EM-18 RFID module (refer Fig. 3) is used to read the RFID tag. It decodes and transmits the signal to Arduino via a serial communication protocol.

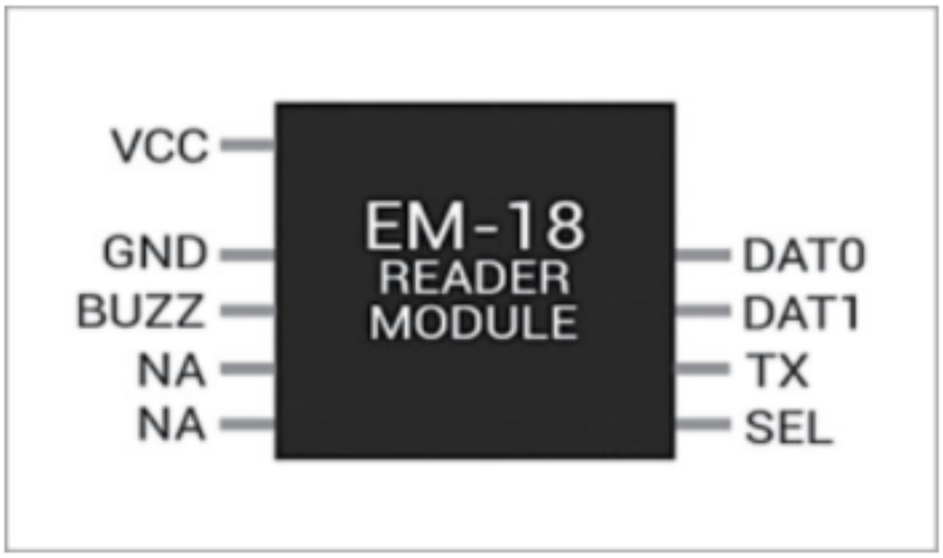


Fig. 3: EM-18 RFID reader

LCD

A 16x2 LCD (liquid crystal display) displays the data received from Arduino board. Solenoid. A 12V solenoid is used to control the flow of liquid. When the solenoid gets energised, it opens its valve and allows fluid to flow through the flow sensor; otherwise it stays closed and does not allow fluid to flow across it. The solenoid valve used in this project is shown in Fig. 4.

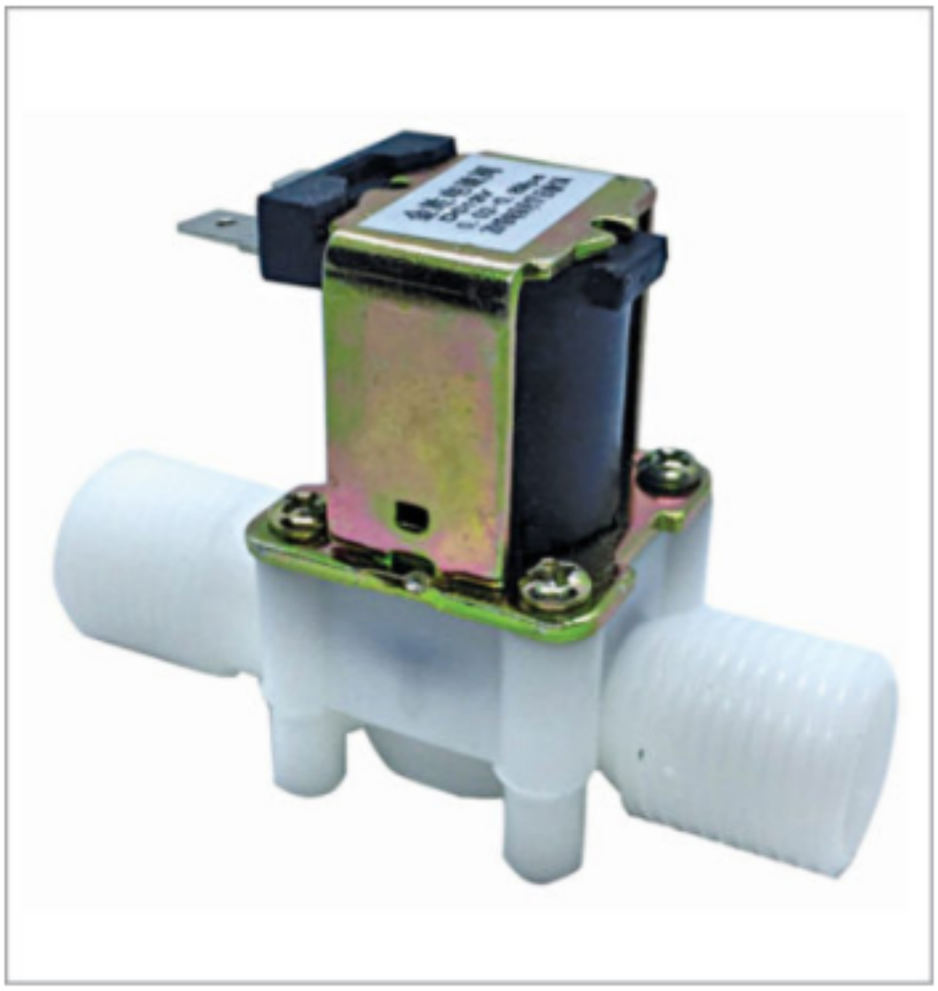


Fig. 4: Solenoid valve

In this project, Arduino is the brain which controls the whole process. Analogue flow sensor connected to the Arduino board is shown in Fig. 5.

When liquid flows through the flow sensor, the liquid pushes against fins of the rotor, causing it to rotate. The shaft of the rotor is connected to a Hall effect sensor. It is an arrangement of a current flowing coil and a magnet connected to the shaft of the rotor, thus a voltage/pulse is induced as this rotor rotates. In this flow meter, for every litre of liquid passing through the sensor per minute, it outputs about 4.5 pulses.

Output pin of this flow sensor is connected to digital pin 2 of Arduino. When liquid flows, the output pulses are counted by the microcontroller in Arduino.

The total number of received pulses is converted into a specific unit like millilitre per second or litre per minute. This is due to the changing magnetic field caused by the magnet attached to the rotor shaft. Here, the flow rate in litres per minute (L/min.) is calculated using a simple conversion formula.

Pin Tx of EM-18 reader is connected to receive pin Rx of Arduino. When RFID tag swipes across EM-18 reader, it sends data to Arduino. If it matches with the programmed data, it sends a signal to the input of the relay driver consisting of transistor BC547 (T1). This input signal makes transistor to conduct, relay RL1 to get energised, which in turn connects 12V supply to solenoid via NO pin of relay and the solenoid gets energised. At the same time, “Place your pot/jug/cup” message is displayed on the first line of LCD1.

Energisation of solenoid allows the liquid to flow through the flow sensor. As the liquid starts to flow through the flow sensor, the amount of liquid being dispensed is displayed on the second line of LCD1.

When a predefined amount (programmed in the code) of liquid has passed through the flow sensor, the relay automatically gets de-energised and the flow of liquid stops. At the same time, “Remove your pot/jug/cup” message is displayed on the first line of LCD1.

If you want the same amount of liquid again, you just swipe the RFID card again across the RFID reader. You can repeat this again and again till the amount of liquid in the container finishes.

Software

The software code (liquid_vending.ino) is written in Arduino programming language. Arduino IDE is used for compiling and uploading the sketch to ATmega328 microcontroller of the Arduino board. Every RFID tag has a unique number. This number has to be included in the Arduino code/sketch.

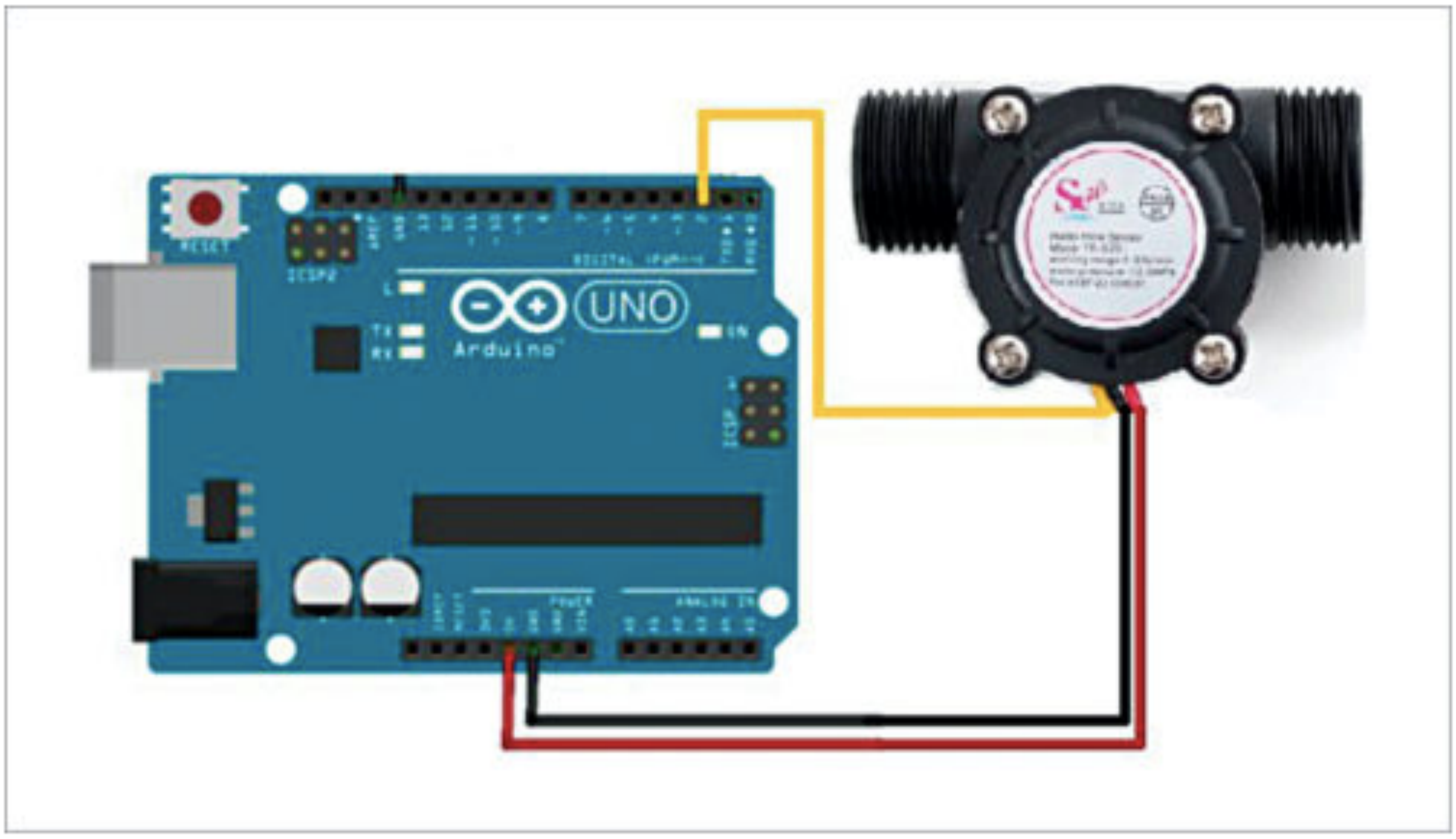


Fig. 5: Analogue flow sensor connected to Arduino

Arduino code uses LiquidCrystal.h header file for the LCD and serial communication for reading the data from the RFID reader. Two built-in functions are used, namely, Serial.available( ) to return the RFID tag number arrived in the serial buffer and Serial.readString( ) to read a string of RFID tag numbers.

In this code/sketch, we used a variable as 'int amount=1000'. This gives a default liquid amount of 1000 millilitres or one litre per card swipe. Just change the amount value as per your requirement.

Download Source Folder: click here

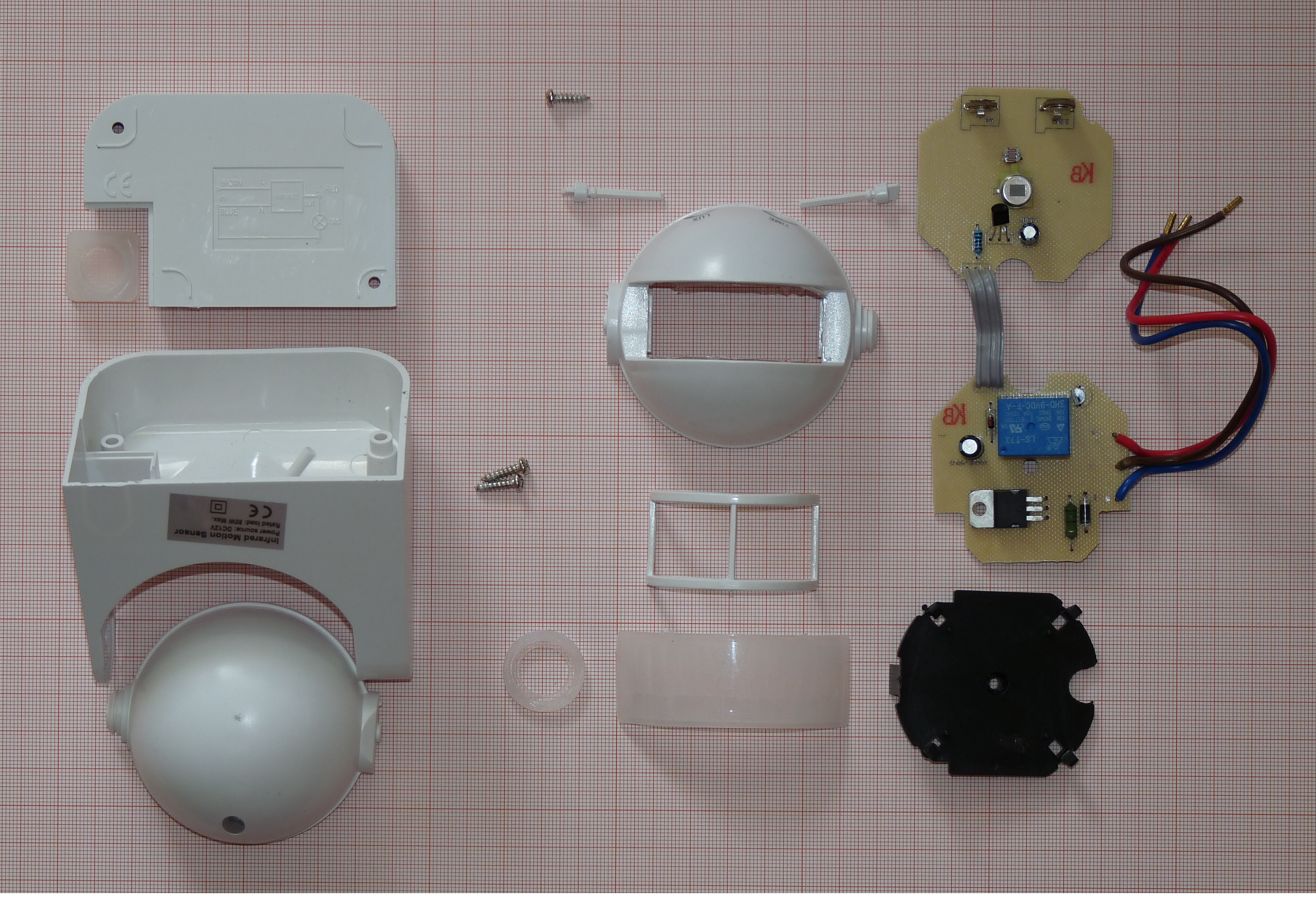
Construction and testing

1. Connect all the components and modules as per the schematic diagram.
2. Connect a 12V power supply from 12V adaptor or 12V battery to Vin pin of Arduino, relay driver and to relay NO pin.
3. Connect relay common pin to the solenoid as shown in the circuit diagram.
4. Check the default amount of liquid in the code for dispensing per minute.

5. Swipe the RFID tag over EM-18 reader. If it matches the programmed RFID tag code, relay and solenoid will be energised, water from the source(tap) will flow through flow sensor.

6. You will get all relevant information on LCD1 as already explained.

Make sure that there is sufficient amount of liquid in the source (water tap or container). Less amount of liquid in the source means less pressure in liquid flow in the pipe, resulting in inaccuracy in the liquid flow measurement.



37

MOTION DETECTOR USING NE555 TIMER

BY KUMAR ABHISEKH

This circuit is based on a passive infrared (PIR) sensor, which automatically switches on a device when someone comes close to it. It can be used for detection of theft or an unauthorised person entering a restricted area or building. It can also turn on lights when someone approaches the area where it is installed. Applications of this circuit include security systems, corridor lights and bathroom lights, among others.

Circuit and working

The circuit diagram of the motion detector using NE555 timer is shown in Fig. 1. It is built around 230V AC primary to 9V, 300mA secondary transformer X1, bridge rectifier DB107 (BR1), 6V voltage regulator 7806 (IC1), timer NE555 (IC2) and a few other components.

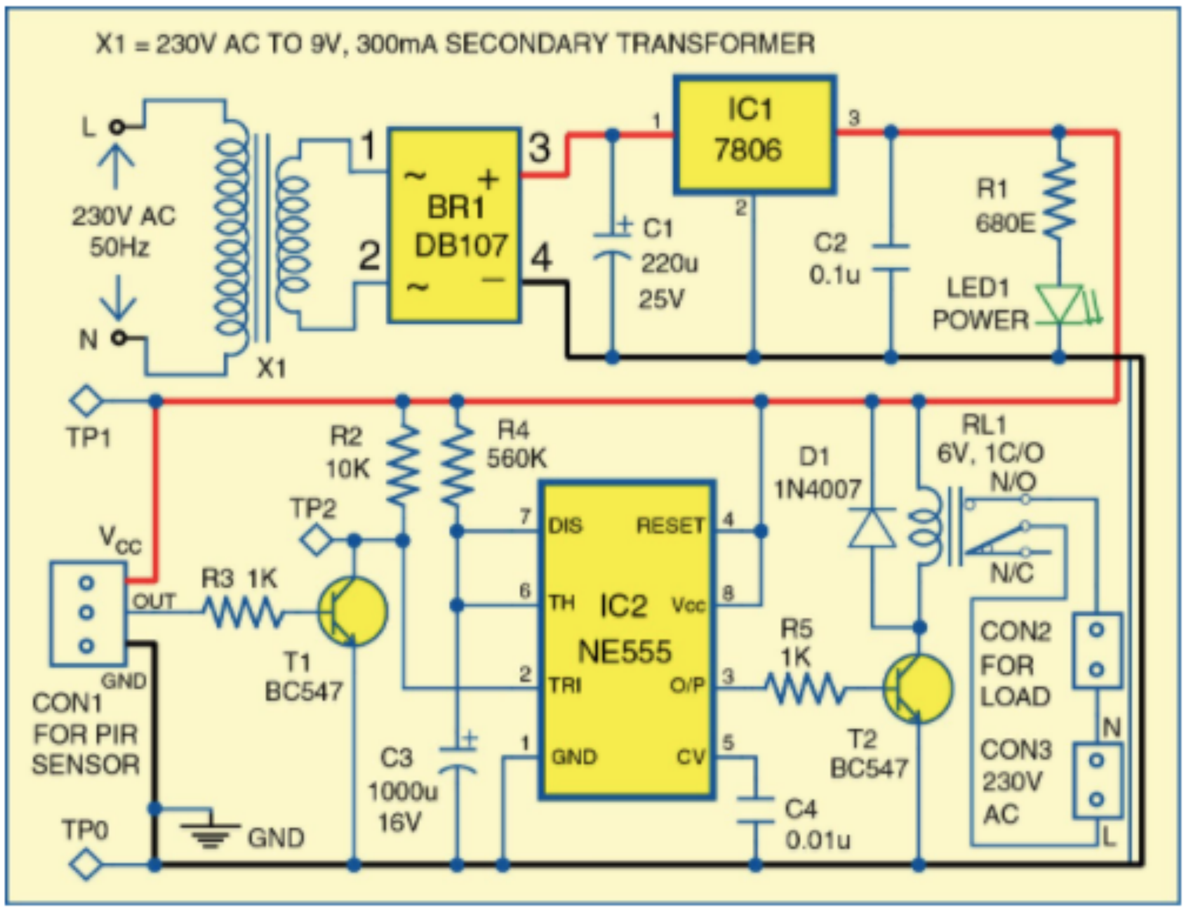


Fig. 1: Circuit diagram of the motion detector

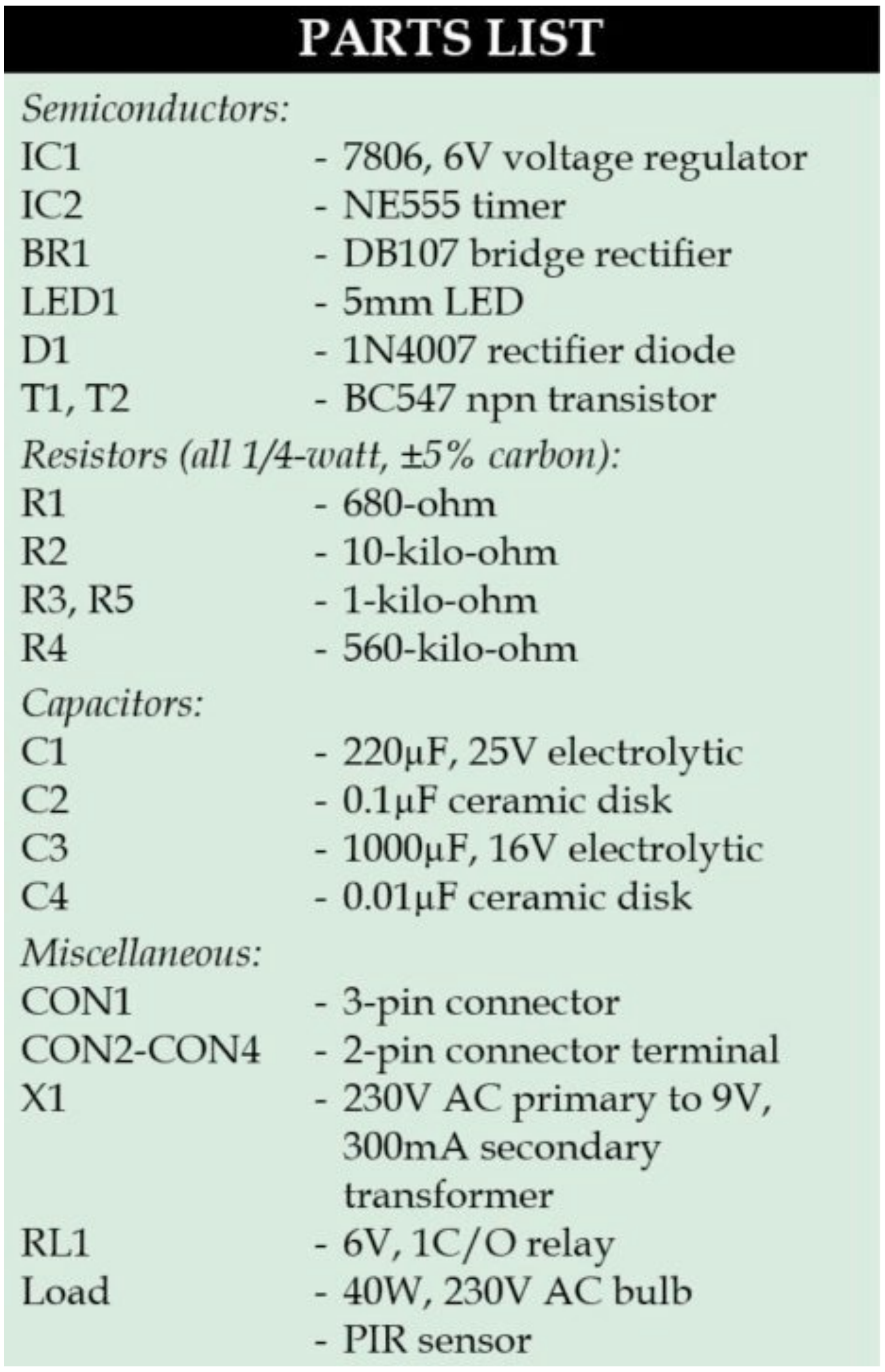
The 230V AC mains is stepped down to 9V, 300mA through step-down transformer X1. Secondary output of X1 goes to bridge rectifier BR1. Capacitor C1 filters the ripples and the rectified output is given to regulator 7806 (IC1). IC1 provides 6V regulated DC output to operate the circuit. LED1 is used as a power-on indicator.

IC2 is configured in monostable mode. Time period of IC2 is based on resistor R4 and capacitor C3, which is around ten minutes in this case. By changing resistor R4 and capacitor C3 you can change the time period of IC2.

Test Points

Test point	Details
TP0	0V (GND)
TP1	6V
TP2	Normally high and becomes low when motion is detected

Output pin 3 of IC2 is connected to the base of relay driver transistor T2. Contacts of relay RL1 are connected to the load, which could be a CFL or a bulb connected across CON2.



PARTS LIST

Semiconductors:	
IC1	- 7806, 6V voltage regulator
IC2	- NE555 timer
BR1	- DB107 bridge rectifier
LED1	- 5mm LED
D1	- 1N4007 rectifier diode
T1, T2	- BC547 npn transistor
Resistors (all 1/4-watt, ±5% carbon):	
R1	- 680-ohm
R2	- 10-kilo-ohm
R3, R5	- 1-kilo-ohm
R4	- 560-kilo-ohm
Capacitors:	
C1	- 220µF, 25V electrolytic
C2	- 0.1µF ceramic disk
C3	- 1000µF, 16V electrolytic
C4	- 0.01µF ceramic disk
Miscellaneous:	
CON1	- 3-pin connector
CON2-CON4	- 2-pin connector terminal
X1	- 230V AC primary to 9V, 300mA secondary transformer
RL1	- 6V, 1C/O relay
Load	- 40W, 230V AC bulb
	- PIR sensor

The PIR sensor is a pyroelectric device developed for detection of human body infrared radiations. It has a single output that goes high when a valid motion is detected. That is, the load is switched on whenever the PIR module senses a body in motion nearby.

Under normal conditions, transistor T1 is cut off and collector output is high around 6V. When motion is sensed, output pin (OUT) of the sensor becomes high, making transistor T1 to conduct for a few seconds, and voltage at its collector goes low momentarily. This low signal triggers IC2. Output pin 3 of IC2 goes high for around ten minutes, energising relay RL1 through transistor T2, turning on the load for ten minutes.

In brief, when someone comes in front of the PIR module, its output triggers IC2 to turn on the load. Thereafter it is disabled automatically.

Construction and testing

An actual-size, single-side PCB for the motion detector using NE555 timer is shown in Fig. 2 and its component layout in Fig. 3. Enclose the PCB in a box and install it at a suitable location. Connect the PIR module across CON1.

Connect transformer X1 to 230V AC mains. Verify various test point voltages as given in the table to ensure proper working before using the circuit.

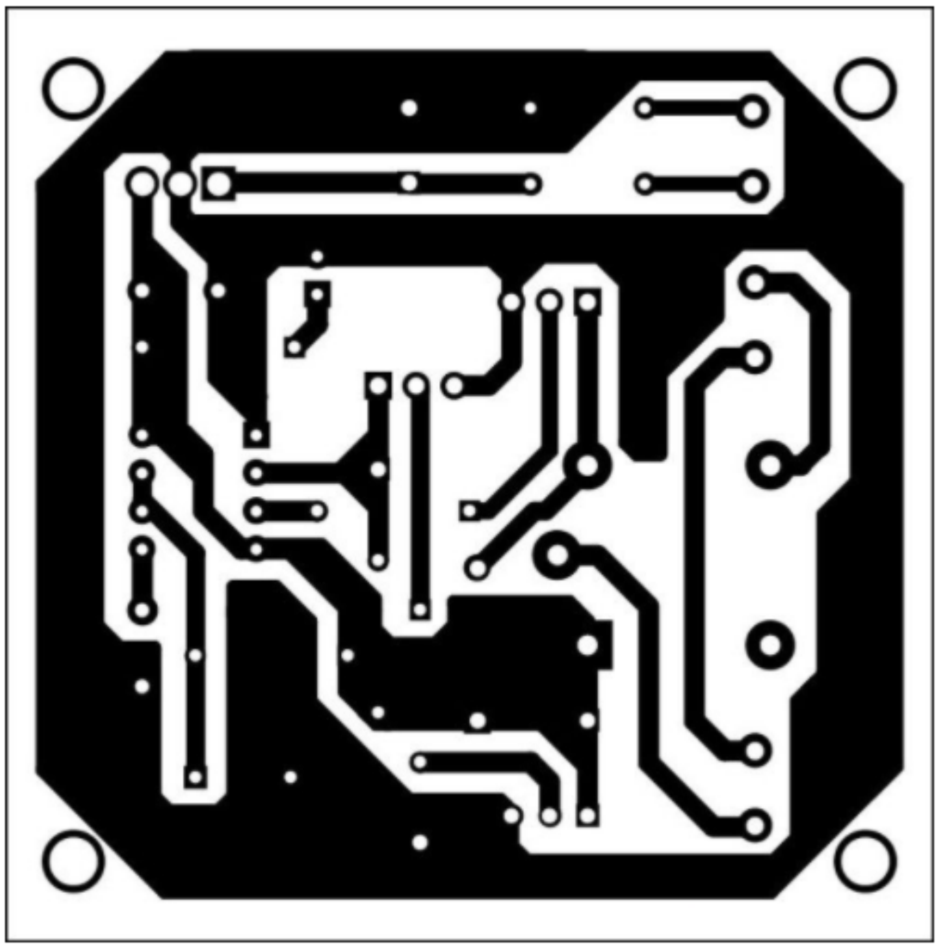


Fig. 2: Actual-size PCB of the motion detector

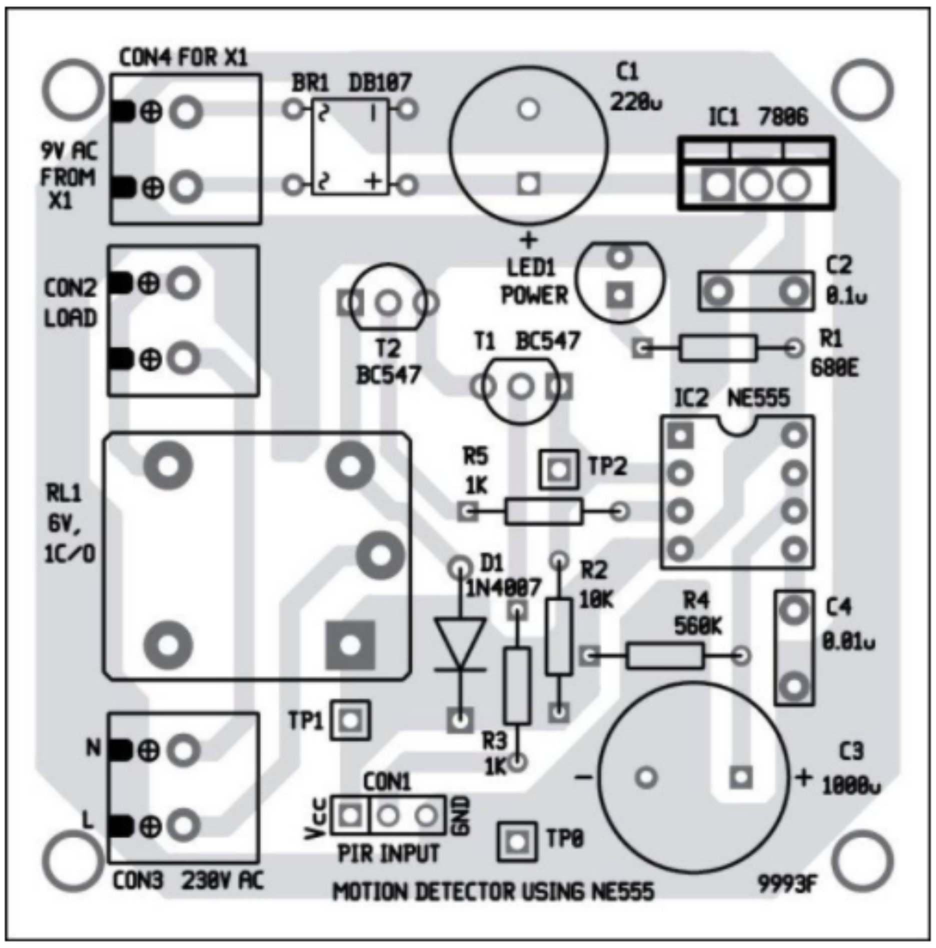


Fig. 3: Component layout of the PCB

Download PCB and Component Layout PDFs: click here

38

RFID BASED ACCESS CONTROL USING ARDUINO

BY AKHIL KAUSHIK

Radio frequency identification (RFID) is used in many applications. Here, we present an access control system based on EM-18 RFID reader module and Arduino UNO board.

This RFID Based Access Control System is built around Arduino UNO board, RFID reader module, solenoid lock and a few other components.

RFID is a non-contact, automatic identification technology that uses radio signals to identify, track, sort and detect a variety of objects including people, vehicles, goods and assets without the need for direct contact or line-of-sight contact (as found necessary in bar code technology).

RFID technology can track movement of objects through a network of radio-enabled scanning devices over a distance of several metres. A device called RFID tag, or simply a tag, is a key component of the technology. These are actively used in RFID based access control system implemented in offices all around.

An RFID system typically consists of three key elements:
1. An RFID tag, or transponder, that carries object-identifying data (unique ID code)
2. An RFID tag reader, or transceiver, that reads and writes tag data
3. A back-end database that stores records associated with tag contents

An RFID reader emits a low-level radio frequency magnetic field that energises the tag.

• The tag responds to the reader's query and announces its presence via radio waves, transmitting its unique identification data.

• This data is decoded by the reader and passed to the local application system via middleware.

• The middleware acts as an interface between the reader and the RFID application system.

• The system then searches and matches the identity code with information stored in the host database or backend system.

In this way, accessibility or authorisation for further processing can be granted or refused, depending on results received by the reader and processed by the database.

RFID based access control System circuit and working

Fig. 1 shows the circuit of RFID based access control using Arduino board. The circuit is built around Arduino UNO board (Board1), RFID reader module (RFID1), solenoid lock and a few other components.

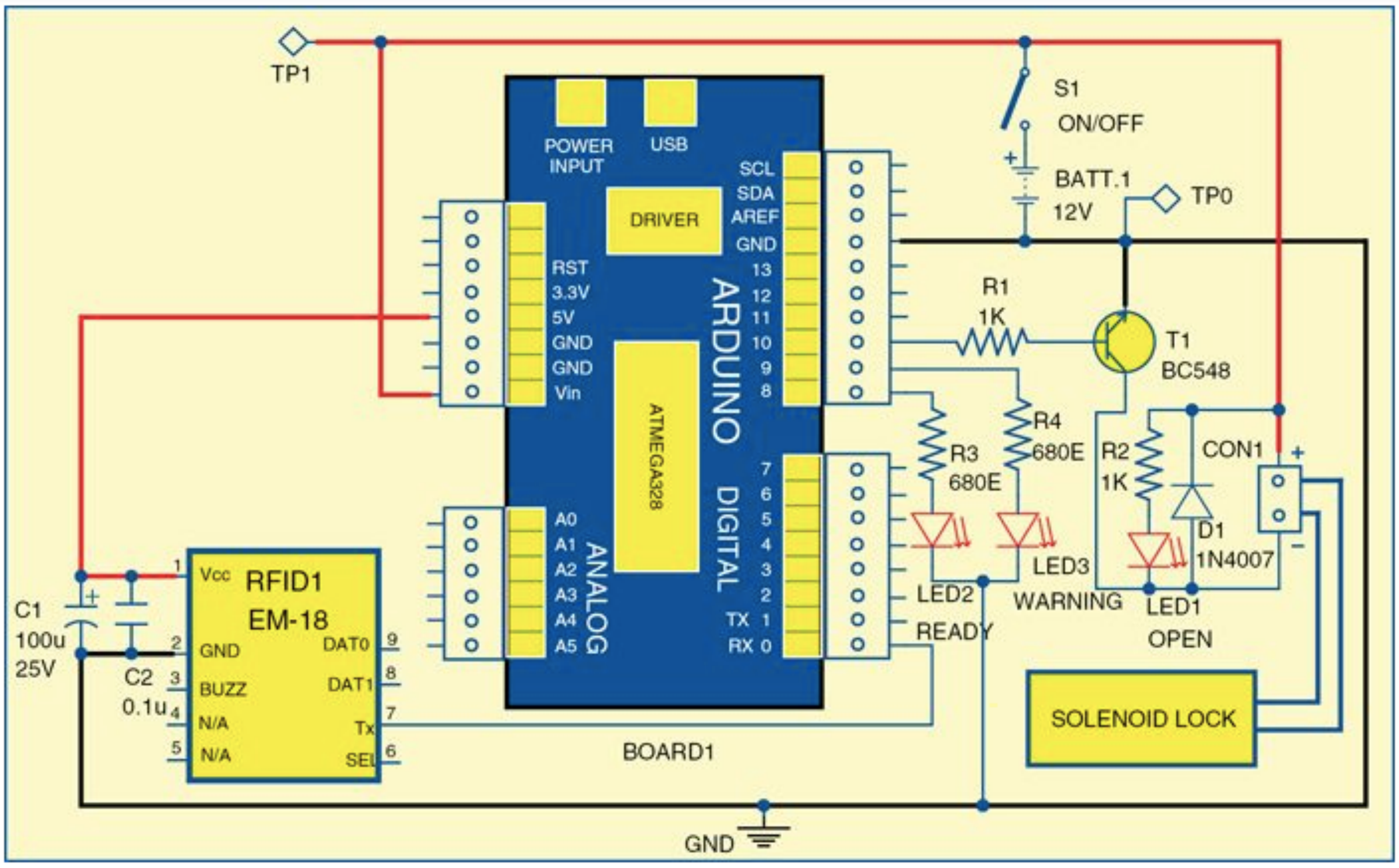


Fig. 1: Circuit diagram of the RFID Based access control system

Arduino UNO board

Arduino is an open source electronics prototyping platform based on flexible, easy-to-use hardware and software (called sketch). It is intended for artists, designers, hobbyists and anyone interested in creating interactive objects or environments.

Arduino UNO is a board based on ATmega328 microcontroller. It consists of 14 digital input/output pins, six analogue inputs, a USB connection for programming the on-board microcontroller, power jack, an ICSP header and a reset button. It is operated with a 16MHz crystal oscillator and contains everything needed to support the microcontroller. It is very easy to use as the user simply needs to connect it to a computer

TABLE I
EM-18 RFID Module Specifications

Heading	Specifications
Reading distance	6-10cm
Dimensions	40mm×20mm×8mm (L×H×W)
Frequency	125kHz
Compatible card codes	Manchester 64-bit, modulus 64 encoding
Current rating	35mA (max.)
Operating voltage	4.6V-5.4V DC

TABLE II
Test Points

Test point	Details
TP0	0V
TP1	12V

with a USB cable or power it with an AC-to-DC adaptor or battery to get started. The microcontroller on the board is programmed using Arduino programming language and Arduino development environment.

Pin 0 (RX) of Board1 is connected to pin 7 (TX) of RFID1. Pin 10 of the board is connected to solenoid driver transistor T1 through base resistor R1. When pin 10 goes high, T1 conducts and solenoid is activated, which means lock is opened.

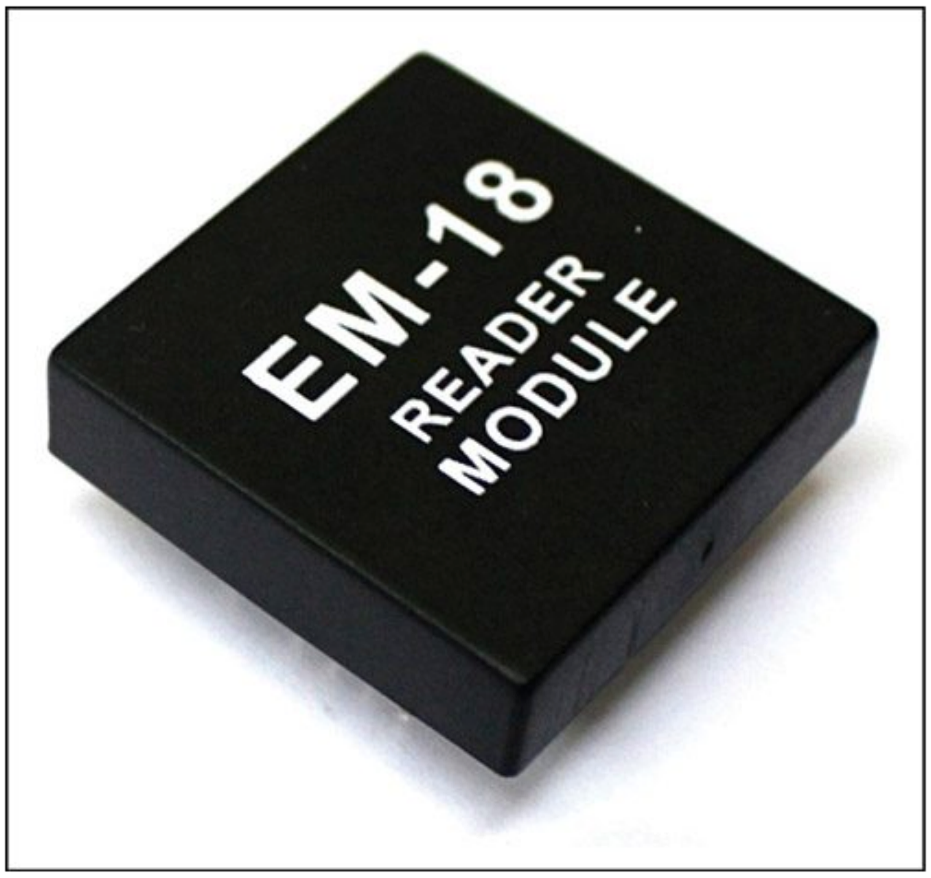


Fig. 2: RFID reader module

RFID reader module

In this project, we used EM-18 RFID reader module (Fig. 2) operating at 125kHz. The module comes with an on-chip antenna and can be powered with a 5V power supply. The transmit pin (TX) of the module should be connected to receive pin (RX) of Arduino UNO board. Some specifications of the module are listed in Table I.

RFID tag

An RFID tag is a smooth card of credit-card size (Fig. 3), which is read by an RFID reader. It works at 125kHz and comes with a unique 32-bit ID. Normally, each tag has a unique ID number which cannot be changed. You can find out its unique ID through software.

Fig. 3: RFID tags

Software

The software for this project is written in Arduino programming language. The Arduino UNO is programmed using Arduino IDE software. Atmega328 on Arduino UNO comes with a boot loader that allows you to upload new code to it without the use of external hardware programmer. It communicates using STK500 protocol.

You can also bypass the boot loader and program the microcontroller through in-circuit serial programming (ICSP) header, but with boot loader the programming is quick and easy.

There are two sketch software codes for this application: ReadTag.ino for reading the tag's unique ID and AccessControl.ino for the main application.

Reading tag ID

To read the tag's unique ID, first you need to upload ReadTag.ino sketch into Arduino board by following the steps given below:

1. Connect TX pin of RFID1 to RX pin of Board1 as shown in the Fig. 1
2. Launch Arduino IDE. Connect Board1 to your PC and select the correct serial COM port and Board from Tools menu. Open ReadTag.ino sketch and compile it using Arduino IDE.
3. Burn ReadTag.ino sketch into the microcontroller in Arduino board
4. Open Serial Monitor from Tools menu in Arduino IDE
5. Hold RFID tag close to RFID1
6. Note down the tag ID shown on the Serial Monitor window. This ID will be used in the sketch later in the main application

Programming the main application

The main application of this project is access control system. So we need to upload AccessControl.ino sketch into the microcontroller in Board1. Note that before uploading the sketch into the microcontroller, you should remove the RFID reader module from the circuit. For programming:

1. Open the AccessControl.ino sketch from the Arduino IDE

2. Change the tag ID in AccessControl.ino sketch with the ID you have noted down earlier
3. Connect the Arduino board (Board1) with the PC
4. Upload the sketch into the board

If uploading is successful, you will see the glowing of LED2. It means the system is ready to read the tag. Now, bring the tag near RFID1 reader. If tag ID matches with the ID in the code, solenoid lock will open for five seconds. It closes automatically after five seconds.

Glowing of LED1 indicates that the lock is open.

Glowing of warning LED3 means that you are using the wrong tag.

Construction and testing

A single-side PCB (Arduino shield type) for access control is shown in Fig. 4 and its component layout in Fig. 5. Assemble the circuit on recommended PCB to save time and minimise assembly errors. Double-check for any overlooked error.

To test the circuit for proper functioning, verify correct 12V supply for the circuit at TP1 with respect to TP0.

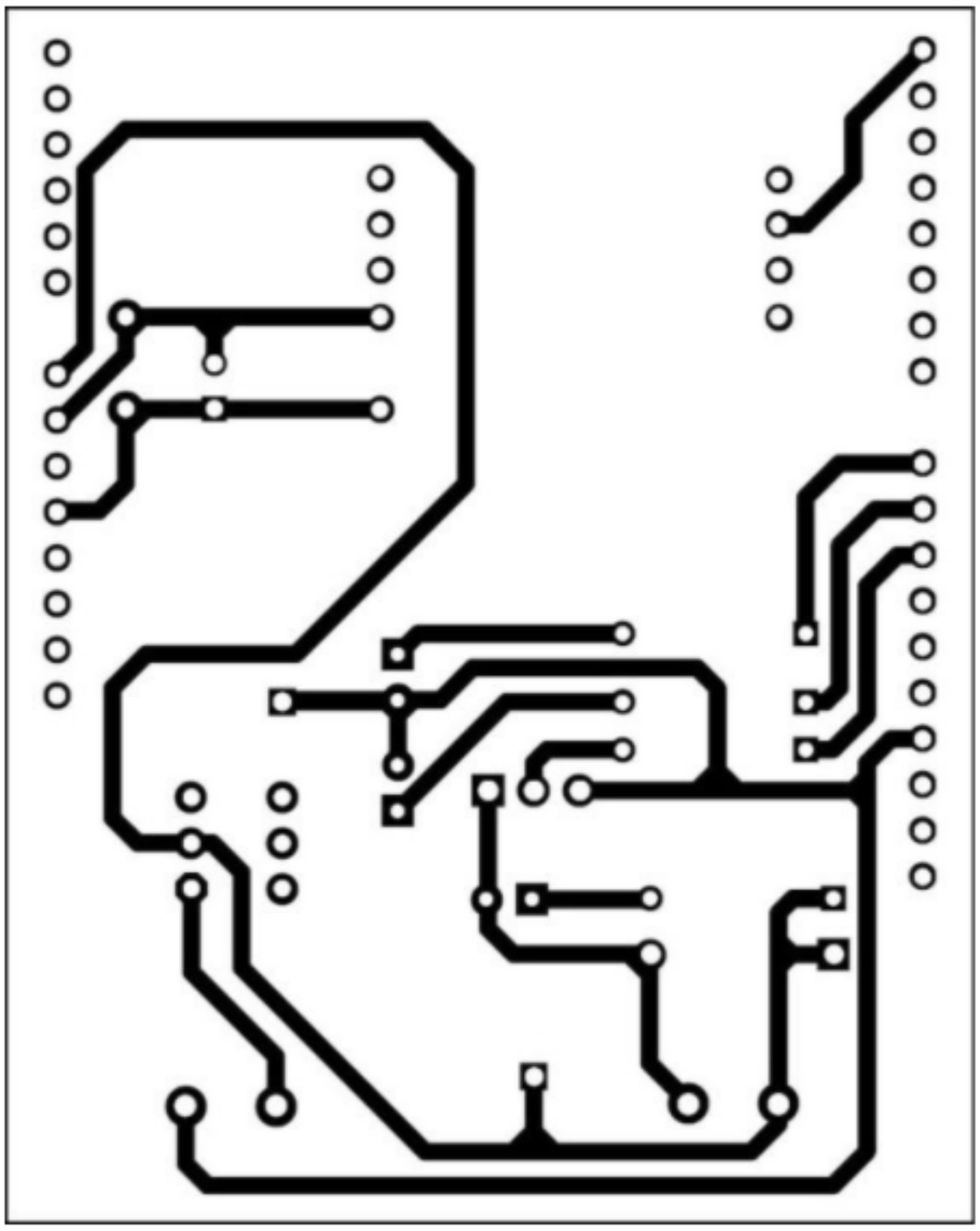


Fig. 4: An single side PCB layout for the circuit

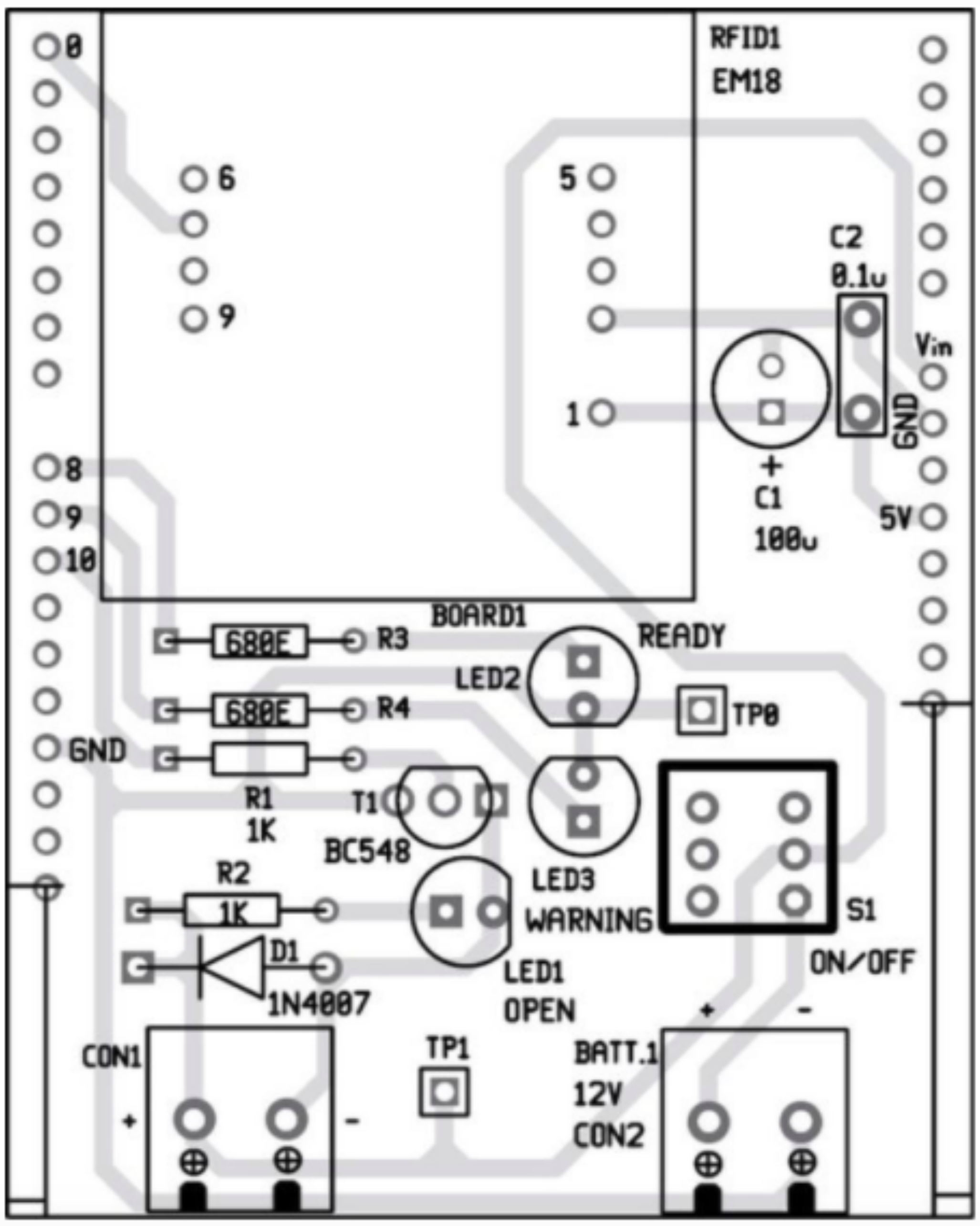


Fig. 5: Component layout for the PCB

Download PCB Layout and component layout PDFs: click here

Download source code: click here

39

SMART GPS GEOFENCING SYSTEM

BY ASHWINI KUMAR SINHA

Geofencing is one of the greatest technologies of the 21st century. It has a multitude of applications such as car tracking and parcel location tracking. Here is a more exhaustive list of the applications of geofencing

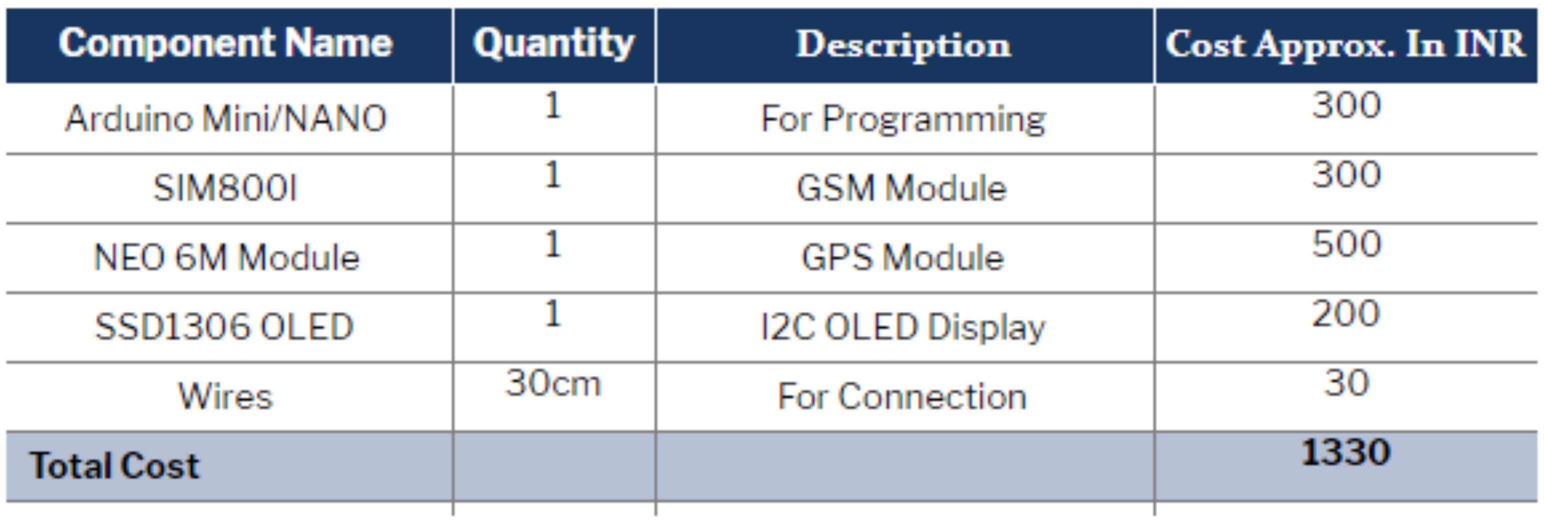
- Can be used for livestock and other animals to track their movement
- Can be used to monitor prisoners or people under house-arrest
- Can be used for the protection of valuable items

- Can be used by parents for child monitoring
- Can be used in rental cars

Geofencing systems are easy to make and have plenty of uses, which is why today we will be making a smart Geosystem. We will be using the SIM800l module and a Buzzer for the Distance Alert and notification system. For geo-location we will use a NEO 6M Gps module. We will also connect a I2C OLED Display that tells us the distance between the base and current location. In this device we will set a base location and a threshold distance. When the device detects that it is moving beyond the prescribed distance, the buzzer attached to pin 13 will automatically start to alert and at the same time, the Sim800l will send a message to set numbers alerting them of the crossing of the set distance.

Let's start our project by shopping for the following components:

Bill Of Materials



Component Name	Quantity	Description	Cost Approx. In INR
Arduino Mini/NANO	1	For Programming	300
SIM800l	1	GSM Module	300
NEO 6M Module	1	GPS Module	500
SSD1306 OLED	1	I2C OLED Display	200
Wires	30cm	For Connection	30
Total Cost			**1330**

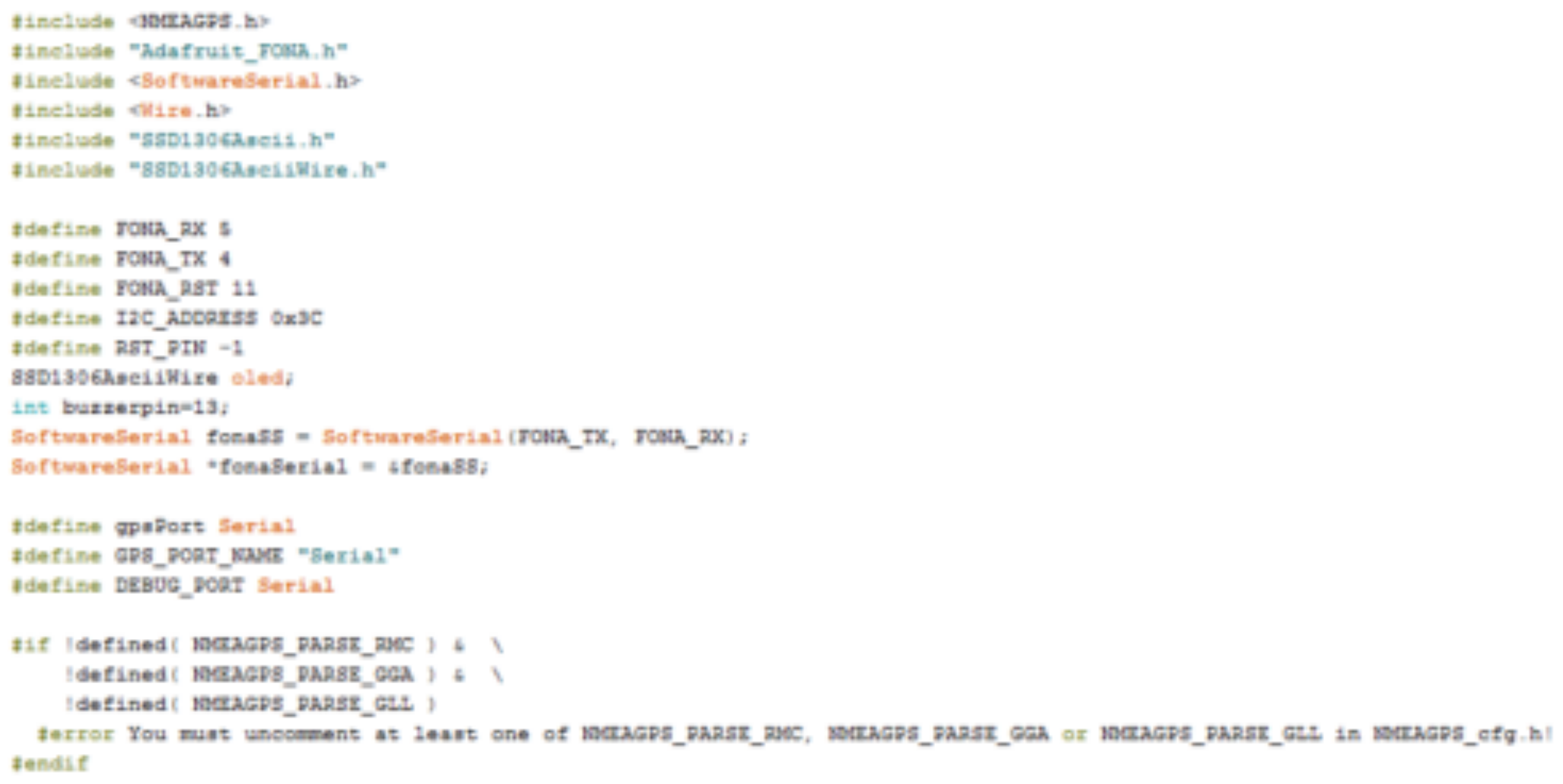
Prerequisites

Assuming that Arduino IDE is already installed in your system, we have to do some basic setup of required library. To do so, go to Sketch menu-->click the manage library option -->in Library manager window search the library ("Adafruit FONA","NMEA GPS" and "SSD1306 ASCII") one by one and install them.

Now after installation of library we can proceed to the coding part.

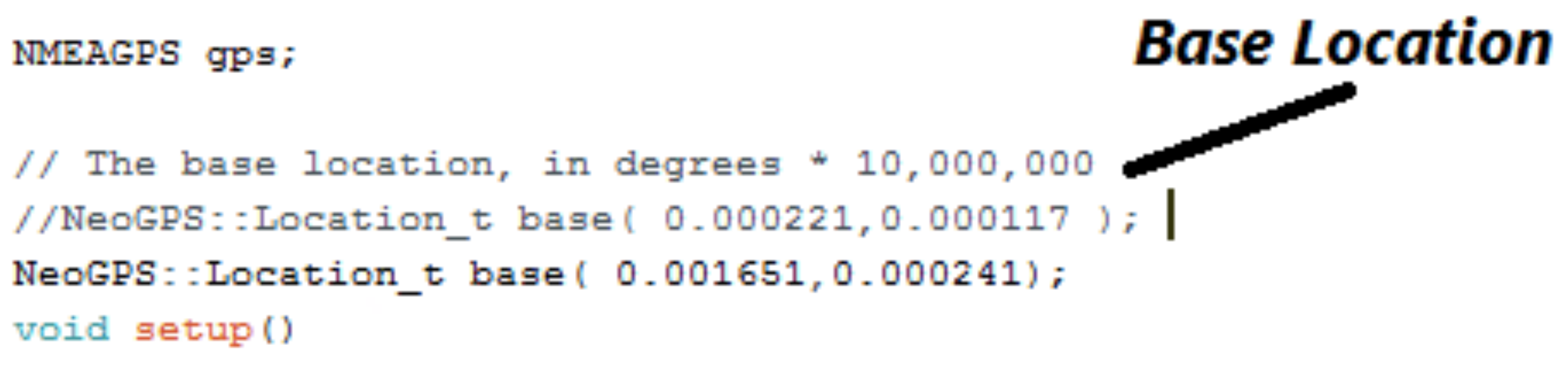
Coding

In the first part of the code we need to initialize the required library (“Adafruit FONA”,”NMEA GPS” and “SSD1306 ASCII”) then we have to define the pin numbers for Serial Communication with SIM 800l Module. After that, we will setup the base location from where we want to start the geo fencing, or, you can say, the center point of Geofencing. (Refer Fig 1, 2).



```
#include <NMEAGPS.h>
#include "Adafruit_FONA.h"
#include <SoftwareSerial.h>
#include <Wire.h>
#include "SSD1306Ascii.h"
#include "SSD1306AsciiWire.h"

#define FONA_RX 5
#define FONA_TX 4
#define FONA_RST 11
#define I2C_ADDRESS 0x3C
#define RST_PIN -1
SSD1306AsciiWire oled;
int buzzerpin=13;
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

#define gpsPort Serial
#define GPS_PORT_NAME "Serial"
#define DEBUG_PORT Serial

#if !defined( NMEAGPS_PARSE_RMC ) &  \
    !defined( NMEAGPS_PARSE_GGA ) &  \
    !defined( NMEAGPS_PARSE_GLL )
  #error You must uncomment at least one of NMEAGPS_PARSE_RMC, NMEAGPS_PARSE_GGA or NMEAGPS_PARSE_GLL in NMEAGPS_cfg.h!
#endif
```

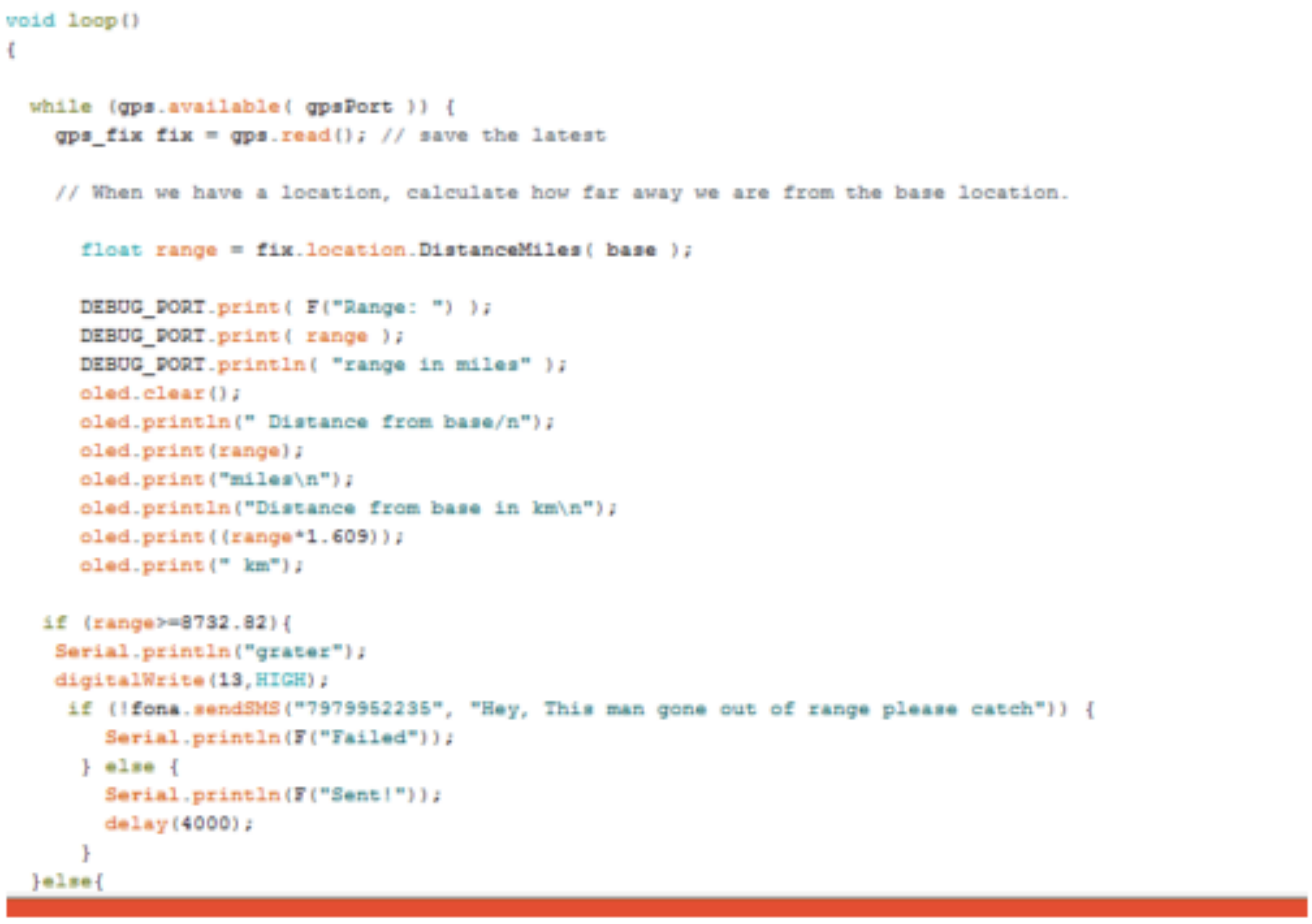
Fig 1



```
NMEAGPS gps;

// The base location, in degrees * 10,000,000
//NeoGPS::Location_t base( 0.000221,0.000117 );
NeoGPS::Location_t base( 0.001651,0.000241);
void setup()
```

Fig 2

Next we will create a setup function where we have set the baud rate for sim800l Module to 4800 that is the default baud rate of our GSM Module. Next, we will set the baud rate for GPS Module as 9600. Here, we have used the hardware serial port of



```
void loop()
{

  while (gps.available( gpsPort )) {
    gps_fix fix = gps.read(); // save the latest

    // When we have a location, calculate how far away we are from the base location.

      float range = fix.location.DistanceMiles( base );

      DEBUG_PORT.print( F("Range: ") );
      DEBUG_PORT.print( range );
      DEBUG_PORT.println( "range in miles" );
      oled.clear();
      oled.println(" Distance from base/n");
      oled.print(range);
      oled.print("miles\n");
      oled.println("Distance from base in km\n");
      oled.print((range*1.609));
      oled.print(" km");

   if (range>=8732.82){
    Serial.println("grater");
    digitalWrite(13,HIGH);
     if (!fona.sendSMS("7979952235", "Hey, This man gone out of range please catch")) {
       Serial.println(F("Failed"));
      } else {
       Serial.println(F("Sent!"));
       delay(4000);
      }
  }else{
```

Fig 4

Arduino for both serial debug and for communication with GPS module. Next we will set the OLED Display and initialise communication with the OLED Display.

Next we will create a loop function that runs repeatedly and checks the GPS Location and updates the location on OLED DIsplay. In the same loop we have created an If condition where we have set the threshold distance or, you can say, the max distance upto which one can be allowed to travel from the base location. If any one crosses this distance, the device sends an alert and also sends a message to set numbers informing them of the geofencing break.

Now, after completing the code, select the Right Port and Board of Arduino and upload it to the Arduino board. After successful upload of code connect all the components as shown in Circuit Diagram in Fig 5.

Connection

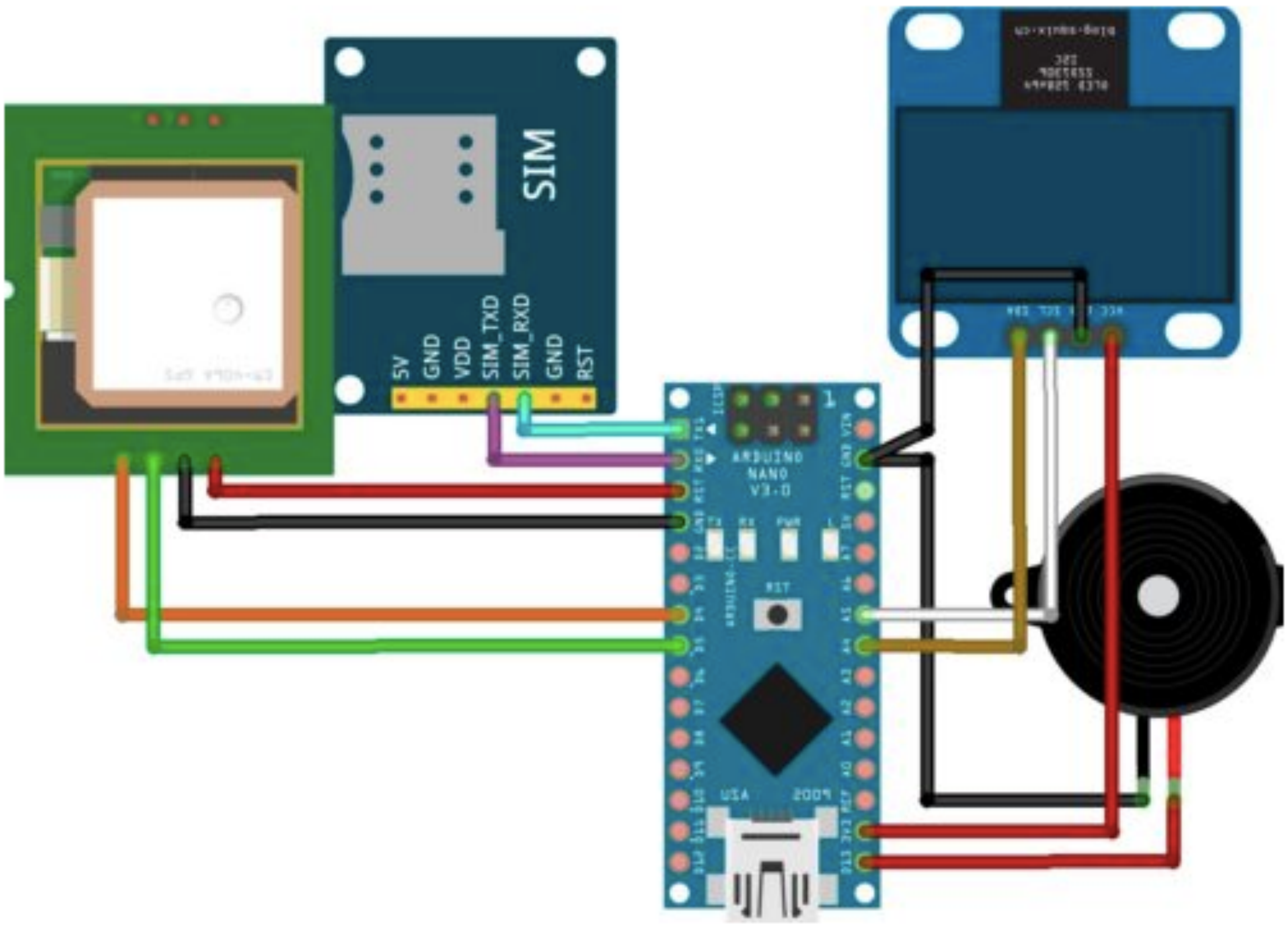


Fig 5. Connection diagram created in Fritzing

Testing

Now after connecting all the components, cross check the connection and then power the arduino board with 5-10 V battery and wait a few minutes. This allows the GPS Module to search the satellite and calculate the location of the device. When the GPS Module gets the location and establishes the connection with the satellite, the red light on the board will start to blink. Now insert a 2G sim in the SIM800l module and wait for a few minutes so that the module registers to the network. Now the display starts showing you the distance away from base location and when you go too far from the base location and cross the threshold distance, the device will automatically alert you and send the message to the phone .

Download Source Code

40

AUDIO MIXER WITH MULTIPLE CONTROLS

BY MALAY BANERJEE

When recording sound from several orchestral instruments being played by different musicians using a single microphone, the only way to adjust the sound balance is to change the position of the musicians relative to the microphone. When recording direct to stereo master tape, it's crucial to make sure that all the voices and instruments sound right before you hit the record button. Presented here is an eight-input audio mixer circuit with bass, treble, volume and balance controls, which you can use to balance sounds from all the sources until you have the desired mix. For capturing the sound from various sources, the audio mixer employs up to eight microphones.

Audio mixer circuit

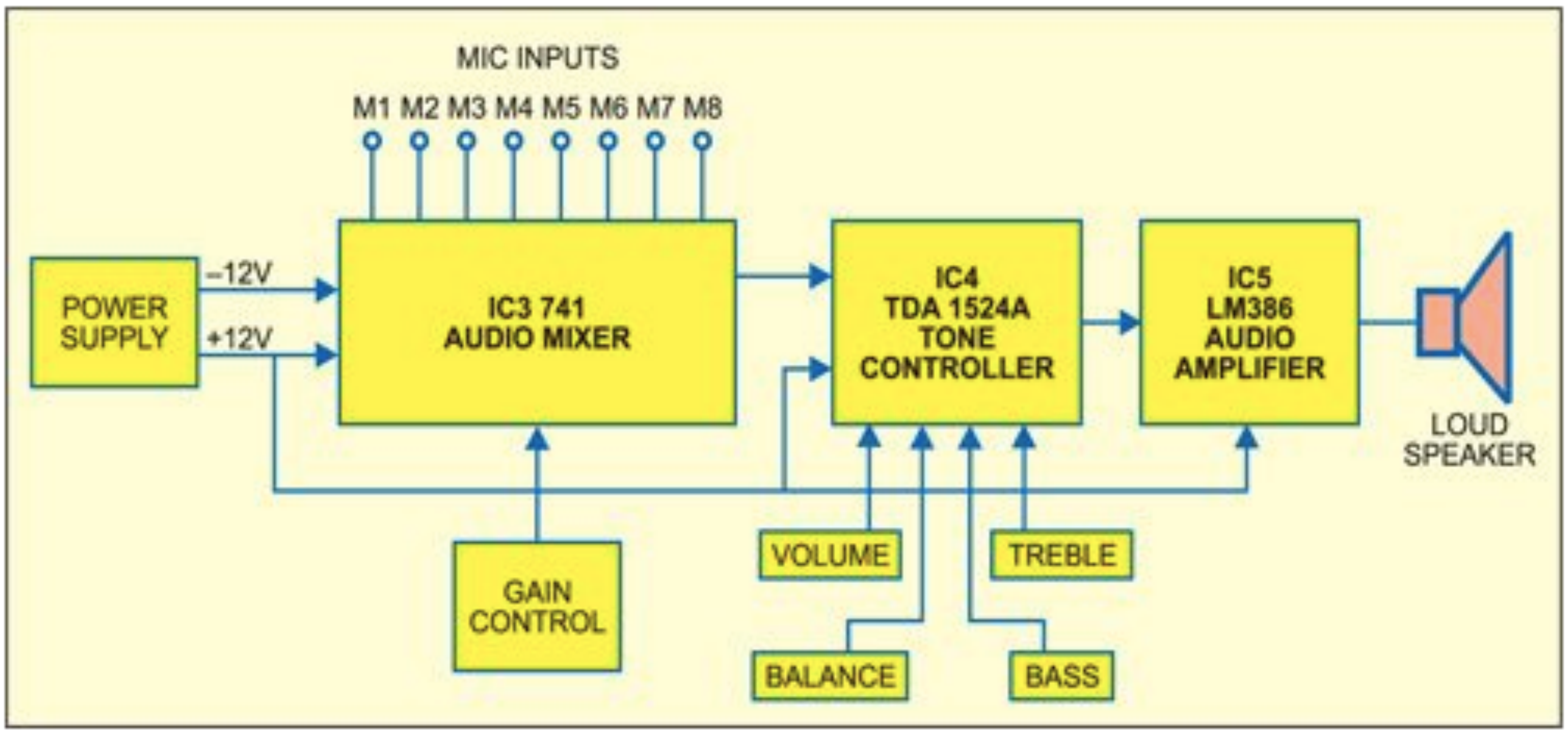


Fig. 1: Block diagram of the audio mixer with bass, treble, volume and balance controls

Fig. 1 shows the block diagram of the audio mixing system along with the audio power amplifier, while the circuit of the audio mixer along with a tone controller is shown in Fig. 2. The power supply and audio power amplifier circuits are shown in Figs 3 and 4, respectively.

Here, dual operational amplifier IC 747 (IC3) is used for mixing several inputs without any mutual interaction. The two internal amplifiers share a common bias network and power supply. The IC has short-circuit protection and wide common-mode and differential voltage ranges.

In this application, +12V and –12V regulated DC supplies are used for operation of IC 747. The microphone output signals M1 through M4, after their individual level adjustments, are mixed and applied across the differential input terminals (pins 1 and 2). Similarly, microphone outputs M5 through M8 are applied across the differential input terminals (pins 7 and 6) of the second amplifier inside op-amp IC 747 after their individual level adjustments.

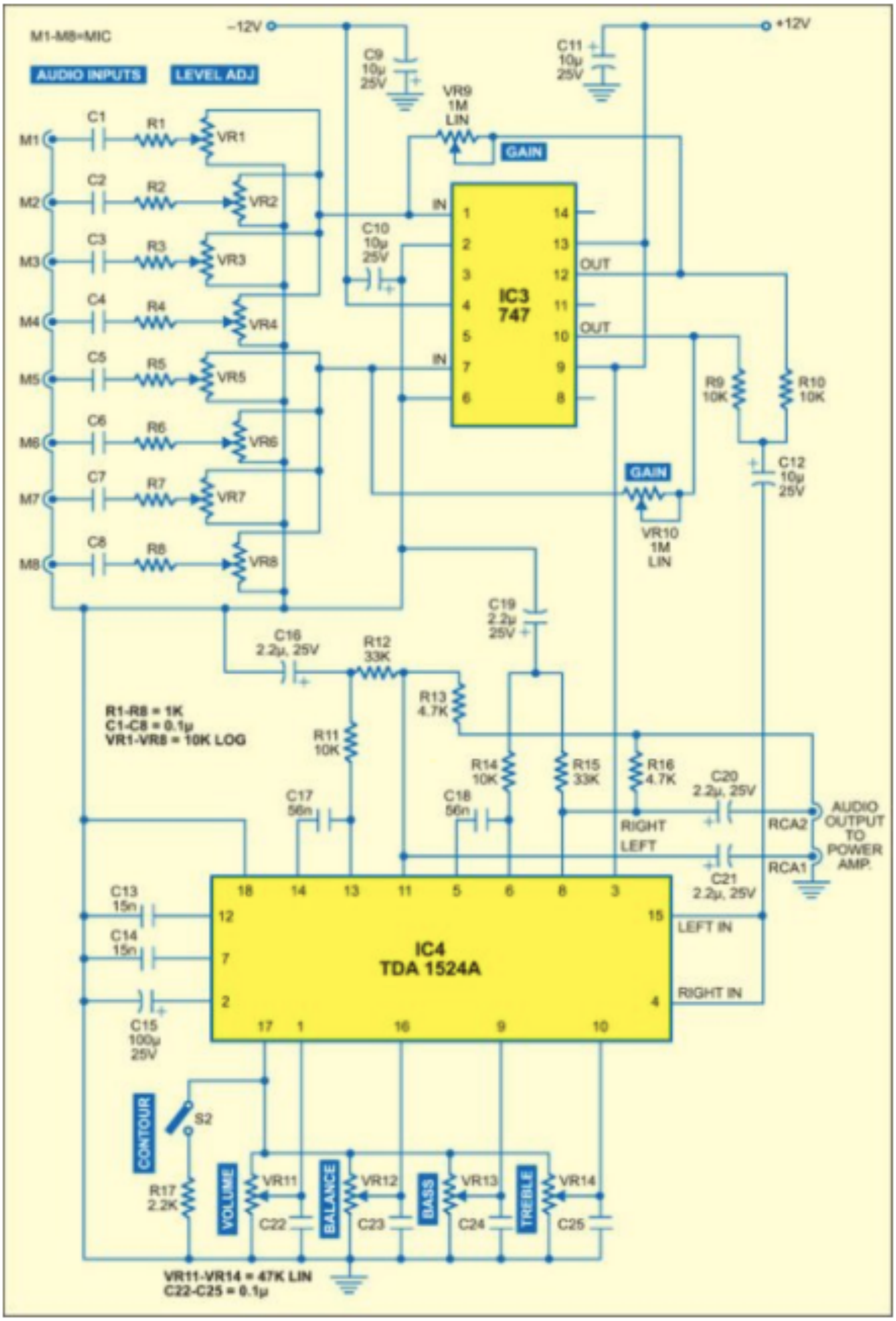


Fig. 2: Circuit of audio mixer with bass, treble, volume and balance control

Circuit Operation

For level adjustment, logarithmic variable resistors VR1 through VR4 and VR5 through VR8, respectively, are employed while feeding the output from respective microphones to the input of the two amplifiers inside IC 747. The outputs of the two amplifiers taken from pins 12 and 10, respectively, are combined at the junction of resistors R9 and R10 before feeding to the next stage (tone controller) via capacitor C12 (10 µF). The overall gain of individual amplifiers can be adjusted with the help of potmeters VR9 and VR10, respectively.

The amplified mixed signal output of IC 747 is applied to shorted input pins 15 and 4 of stereo tone controller IC TDA1524A (IC4). TDA1524A is designed as an active stereo-tone/volume control for car radios, TV receivers and mains-fed equipment. It includes functions for bass and treble control, volume control with built-in contour (can be switched off) and balance. All these functions can be controlled by DC voltages or by single linear potentiometers. This IC serves as an efficient tone controller. Although it may work reasonably well with 9V DC supply, for better bass response, a 12V supply can be used. A good heat-sink is necessary for longer life and better performance of the IC.

Features of TDA1524A are:

1. Simple construction
2. Low noise and distortion
3. Switchable contour (for quick changing of the tonal response)
4. Its output can drive most power amplifiers.
5. Bass emphasis can be increased by a double-pole, low-pass filter
6. Wide power supply voltage range

General specifications are:

1. DC input: 12V (typical)
2. DC battery: 35 mA
3. Maximum output: 3V RMS
4. Maximum input: 2.5V
5. Maximum gain: 21.5 dB
6. Volume control range: –80 to+121.5 dB
7. THD at 1 kHz: 0.3%

8. Ripple rejection at 100 Hz: 50 dB

Potmeters VR11, VR12, VR13 and VR14 are meant for adjustment of volume, balance, bass and treble, respectively. Switch S2 is contour switch, which can be used to change the tonal response of the of the IC. The outputs are available at pins 8 and 11 for right and left channel, respectively. (EFY note. Since both the left- and right-channel input pins 15 and 4 have been shorted in this application, the IC acts as a mono volume/tone control circuit.)

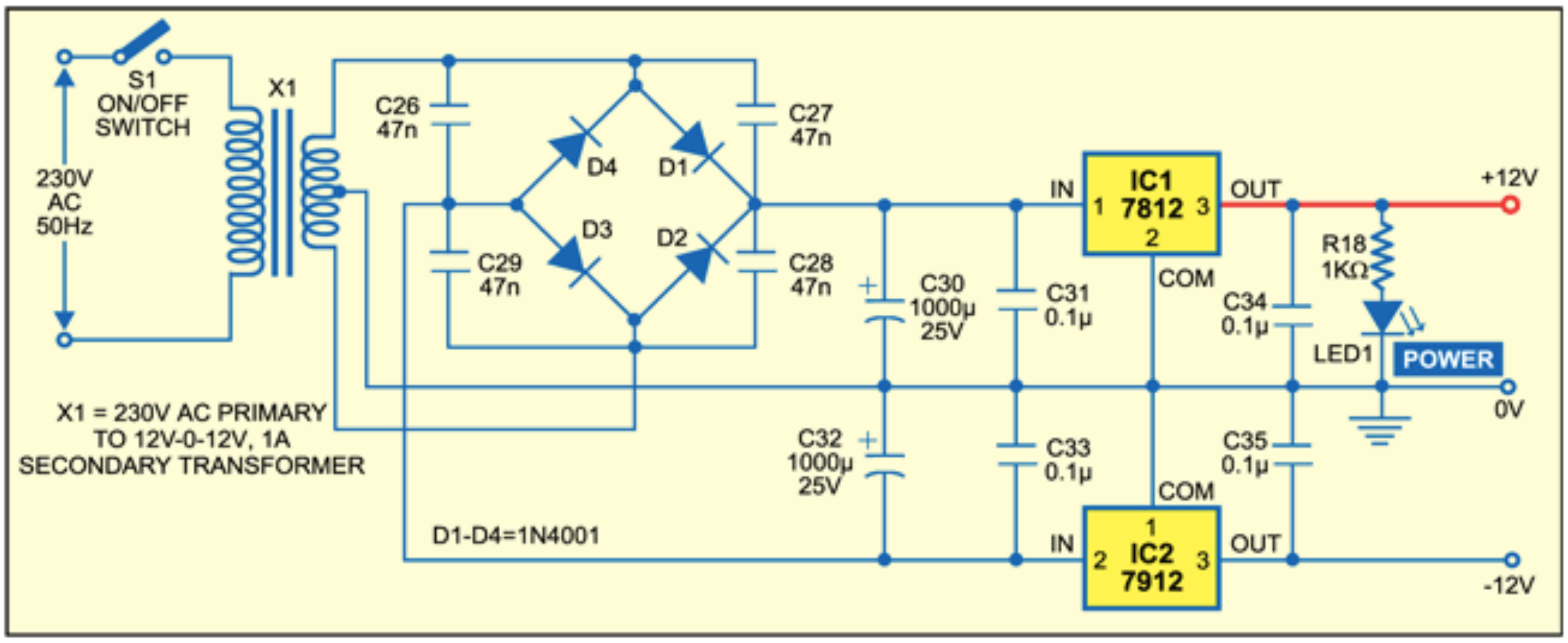


Fig. 3: Power supply circuit

Audio Power Amplifier

The audio amplifier circuit shown in Fig. 4 is optional. One can use much higher-power audio amplifier along with the audio mixer circuit.

The low-power audio amplifier employing IC LM386 (IC5) shown in Fig. 4 can output a maximum audio power of 1 watt. It gets +12V DC

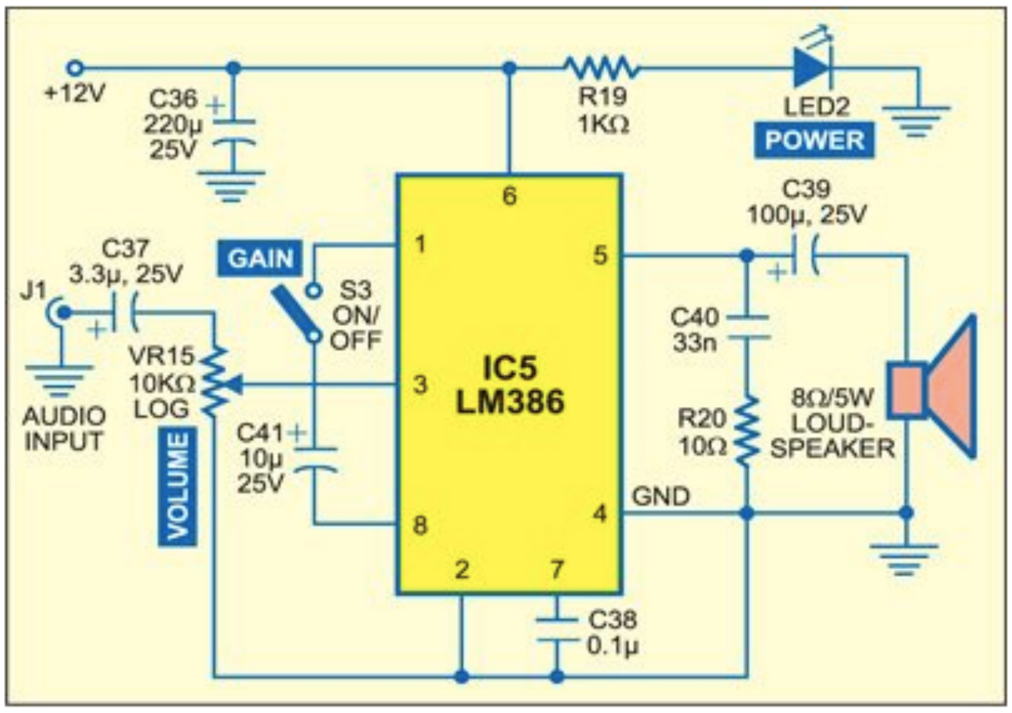


Fig. 4: Low-power audio amplifier circuit

supply at its pin 6. The audio input from sources like Walkman and audio mixer can be fed to pin 3 of IC5 through volume control VR15.

The gain of LM386 is internally set to 20 to keep the external part count low. However, to make LM386 a more versatile amplifier, pins 1 and 8 are provided for setting the gain—externally to any value between 20 and 200—by using an appropriate combination of a resistor and a capacitor. If only a capacitor is put between pins 1 and 8 using switch S3 as shown in Fig. 4, the gain would increase to 200 (46 dB). The amplified output is taken from pin 5 and fed to the loudspeaker through electrolytic capacitor C39 (100 µF). The higher the value of C39, the higher the pitch of the audio frequency response in the speaker.

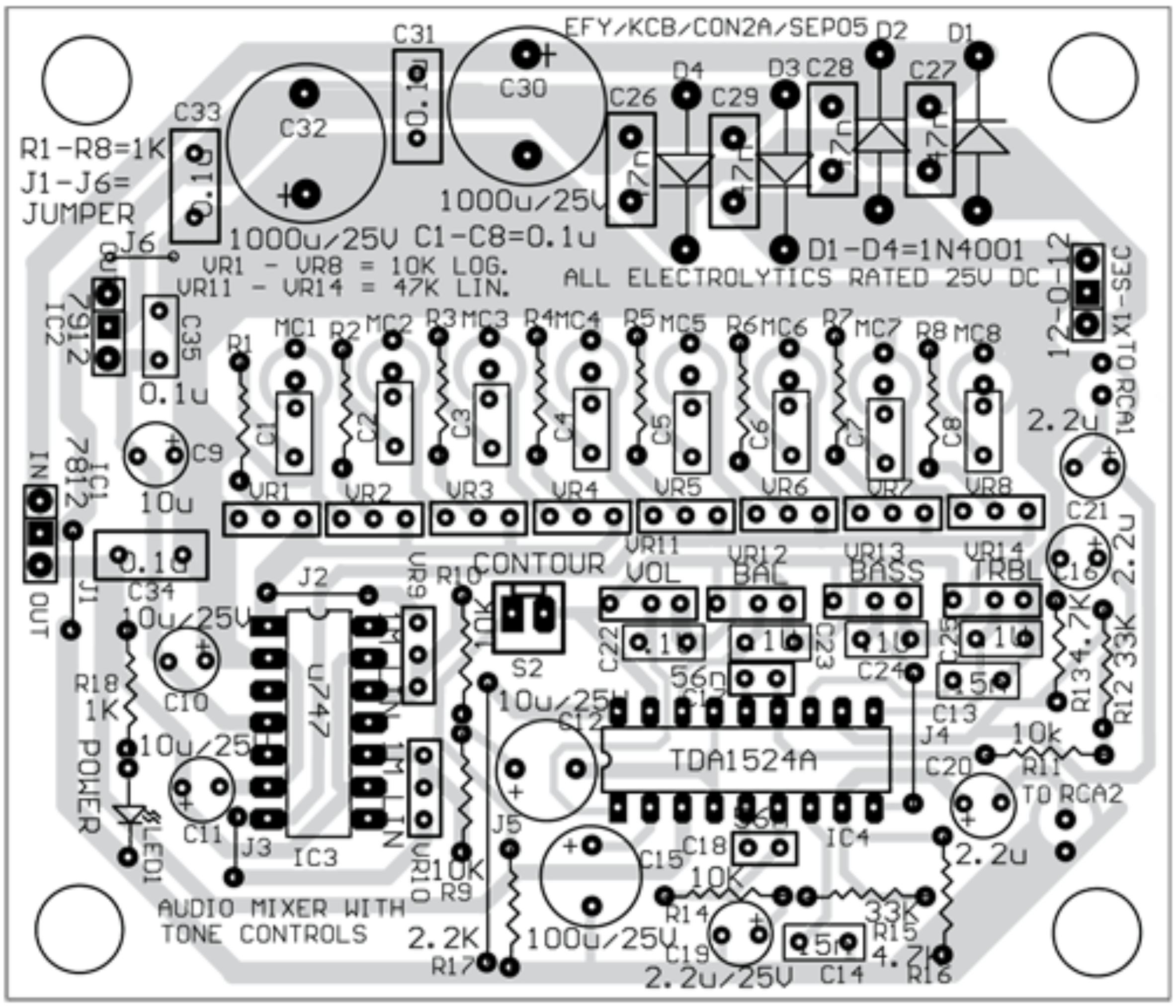


Fig. 6: Components layout for the PCB in Fig. 5

Power supply

The power supply section for the circuit is shown in Fig. 3. It consists of a step-down transformer (230V AC primary to 12V-0-12V, 1A secondary), bridge rectifier, filter network and regulator ICs 7812 and 7912 to provide +12V and –12V regulated DC outputs, respectively. When switch S1 is closed, the presence of power is indicated by the glowing of LED1.

Construction

Assemble the circuit on any general-purpose PCB. Mount IC bases on the PCB. There is no soldering method that is ideal for all IC packages. The use of IC bases prevents damage to the ICs while soldering and also makes it easy to replace them. Use audio input jack connectors for M1 through M8 input points. Also use audio output connectors at the outputs of IC4.

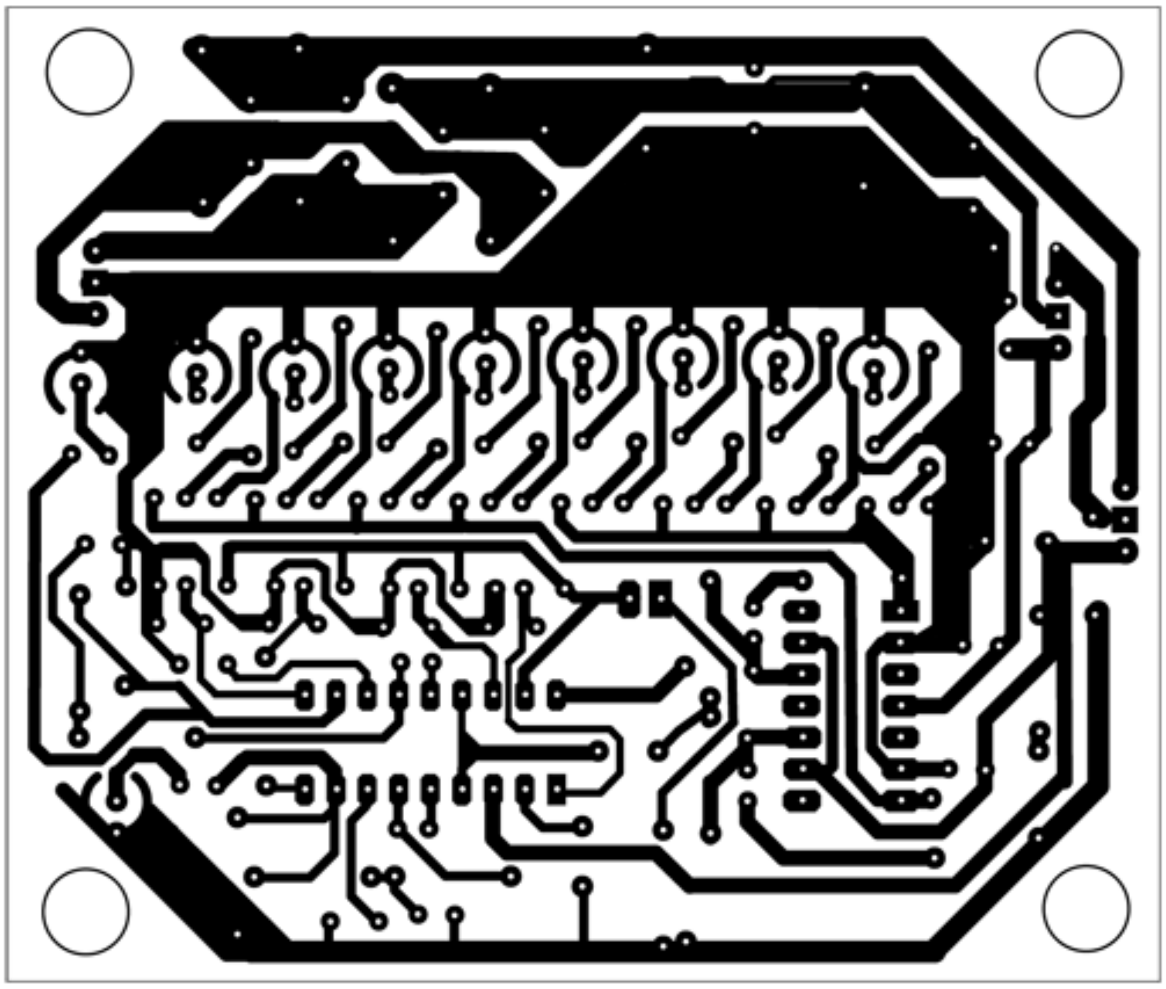


Fig. 5: Combined actual-size, single-side PCB for audio mixer and power supply circuits

Download PCB and component layout PDFs: click here

A combined actual-size, single-sided PCB layout for Figs 2 and 3 is shown in Fig. 5 and its components layout in Fig. 6. The solder-side PCB layout for Fig. 4 is shown in Fig. 7 and its components layout in Fig. 8.

Note

If you are not using IC base for TDA1524A, the maximum permissible temperature of the solder is 260°C; solder at this temperature must not be in contact with the joint for more than five seconds. The total contact time of successive solder waves must not exceed five seconds while using wave soldering.

Testing procedure

1. After assembling the PCB, check the circuit connections before switching on the power supply.

2. Use a standard microphone at the first input point M1 and then keep it near an audio source. You can use the power amplifier circuit given here for testing or another higher-output power amplifier.

3. Vary VR1 slowly until a clear and distortion-free amplified output is obtained.

4. If the sound output is not clear and VR1 does not help, vary gain control VR9.

5. If the problem still persists, check volume, balance, bass and treble controls.

6. Check the various controls in your audio power amplifier section.

7. Repeat steps 2 through 5 for the rest of the inputs. Having checked all the inputs, now the audio mixer is ready for use.

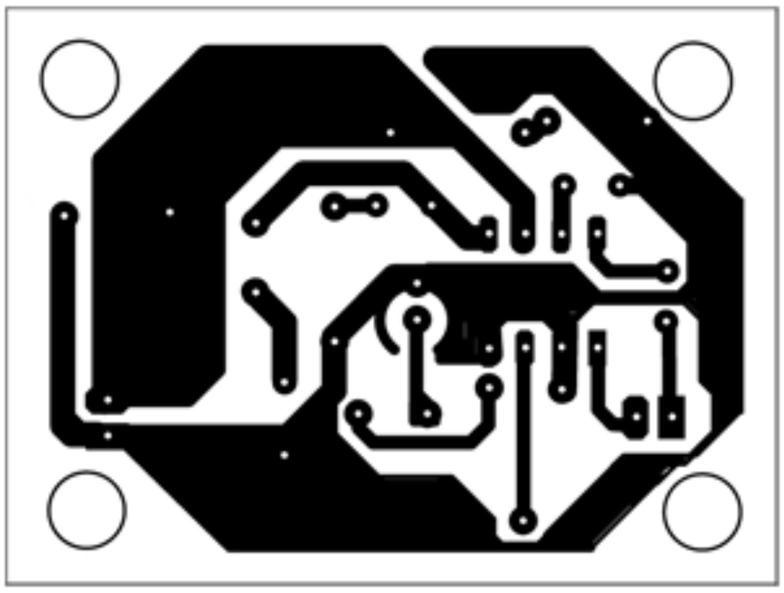


Fig. 7: Solder-side PCB layout for the audio amplifier circuit

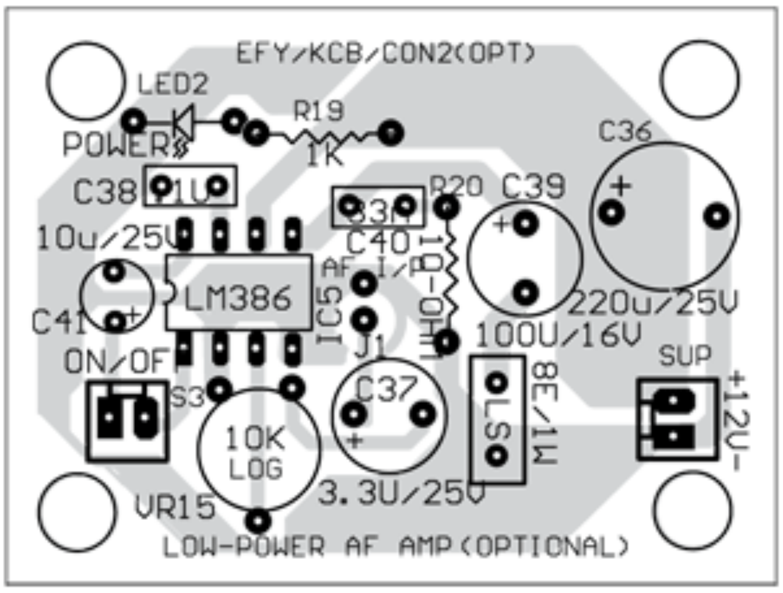


Fig. 8: Components layout for the PCB in Fig. 7

41

NETWORK STORAGE WITH RASPBERRY PI

BY POOJA JUYAL

An always-on network backup device accessible to a computer from both outside and inside the network is a highly desirable thing. As the system will always be on, power consumption is an important consideration. These requirements are fairly met by this Raspberry Pi-based network storage system as it consumes less power than its off-the-shelf counterparts.

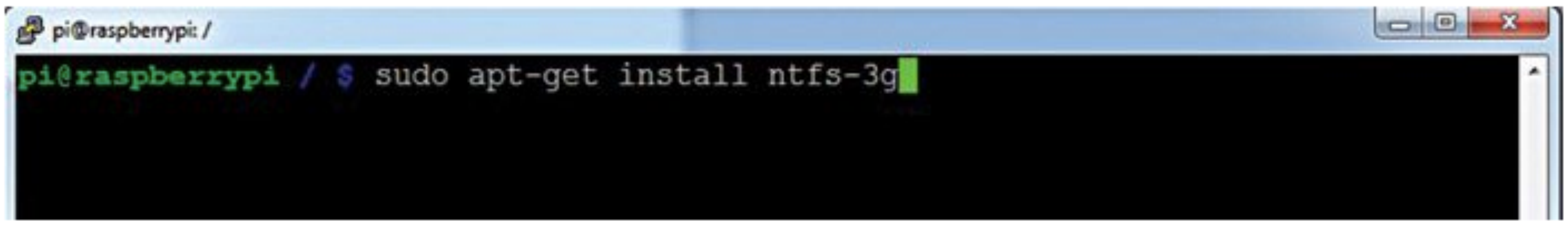
Presented here is a system that uses two USB hard drives plugged into Raspberry Pi to store data. One of the hard drive stores data directly and the other is used to back up the data.

For a simple network, one drive is sufficient but it is recommended to use two for local redundancy. External hard drives should be low-power and have sufficient capacity for future addition of data. In this case, two NTFS-formatted hard drives are used and samba software helps to implement network sharing. Follow the steps mentioned below in the exact sequence to set up and configure the network storage system with Raspberry Pi.

Mounting the external hard disk

Connect the hard drives to the Raspberry Pi and power it up. It is a good idea to use a USB hub with separate power supply. The Raspberry Pi should have a working operating system (Raspbian) configured suitably for remote access to follow all the steps. You can refer ‘Remote Access to Raspberry Pi’ published in June 2013 issue.

Now access Raspberry Pi using SSH and run following commands to add the support for NTFS-formatted drives to Raspbian operating system as shown in Fig. 1:



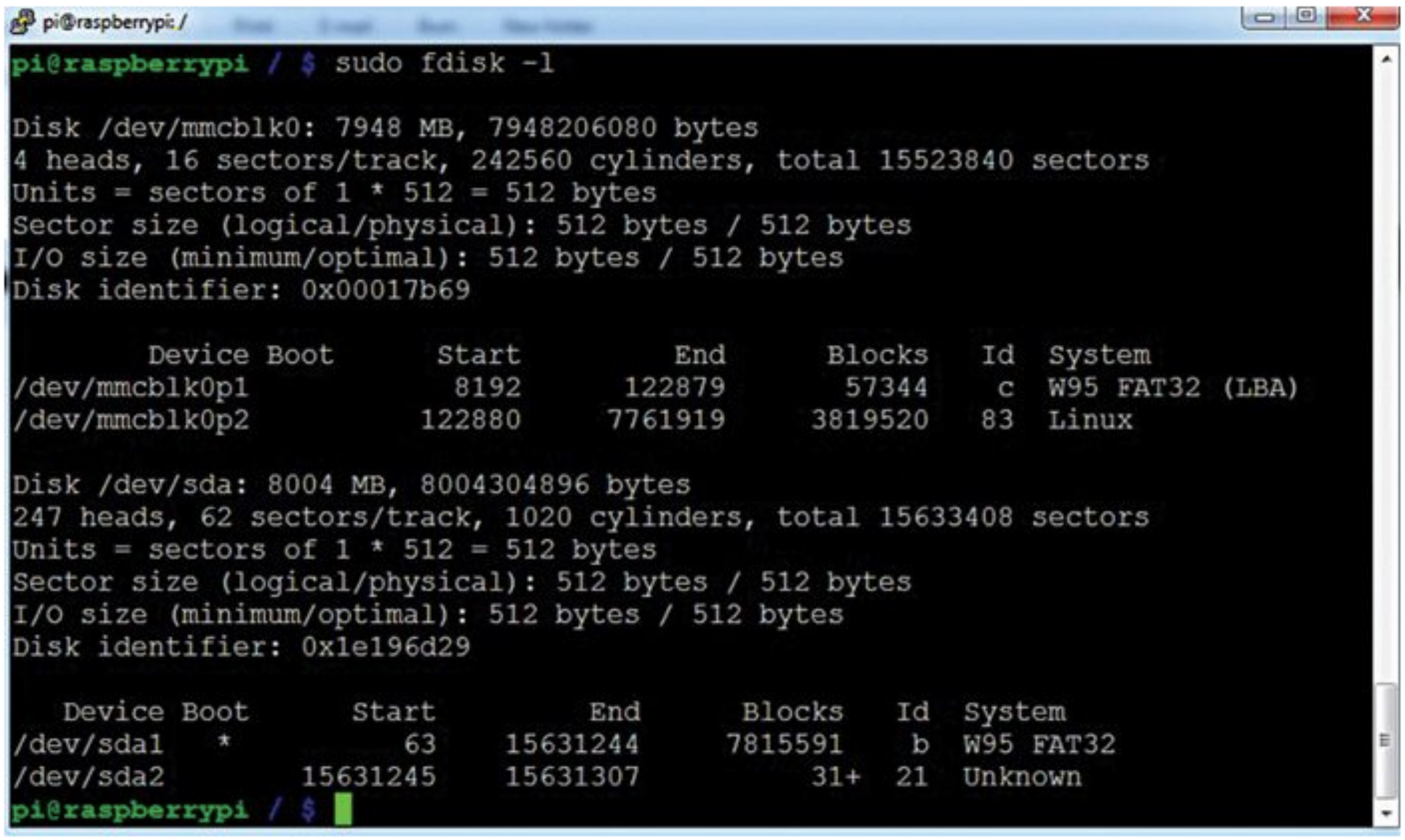
```
pi@raspberrypi / $ sudo apt-get install ntfs-3g
```

Fig. 1: Add NTFS support

$ sudo apt-get
install ntfs-3g

It will take a few minutes for the package to download, unpack and install. Once the NTFS package is installed, look for the un-mounted partitions of the attached external hard drives by executing the following command as shown in Fig. 2:

$ sudo fdisk –l

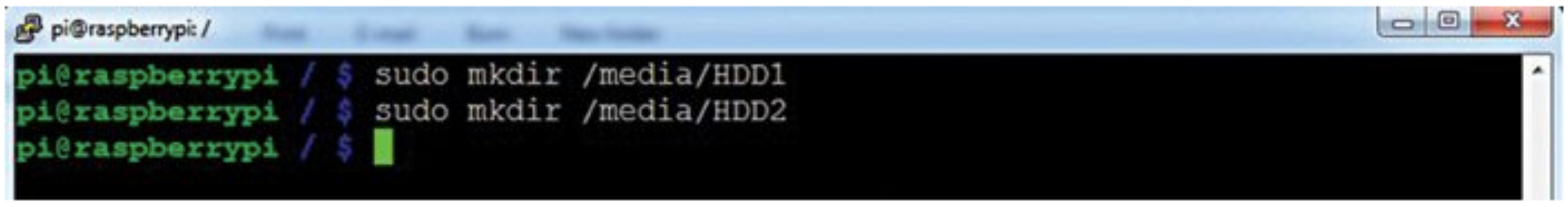


```
pi@raspberrypi / $ sudo fdisk -l

Disk /dev/mmcblk0: 7948 MB, 7948206080 bytes
4 heads, 16 sectors/track, 242560 cylinders, total 15523840 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00017b69

        Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1            8192      122879       57344    c  W95 FAT32 (LBA)
/dev/mmcblk0p2          122880     7761919     3819520   83  Linux

Disk /dev/sda: 8004 MB, 8004304896 bytes
247 heads, 62 sectors/track, 1020 cylinders, total 15633408 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x1e196d29

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *          63    15631244     7815591    b  W95 FAT32
/dev/sda2        15631245    15631307          31+  21  Unknown
pi@raspberrypi / $
```

Fig. 2: Partition of the external hard drives

If only one external hard drive is connected, you will see two disks—one external hard disk and one SD card inside Raspberry Pi. In case two external hard drives are connected, you will see three disks. Carefully make note of the hard drives' names.

Before mounting the drives, we need to create a directory to mount the drives to. For the sake of simplicity, let us create a directory called HDD1 and HDD2 for each drive. To create these directories, write the commands mentioned below and shown in Fig. 3:



```
pi@raspberrypi / $ sudo mkdir /media/HDD1
pi@raspberrypi / $ sudo mkdir /media/HDD2
pi@raspberrypi / $
```

Fig. 3: Make the directories to mount the drives

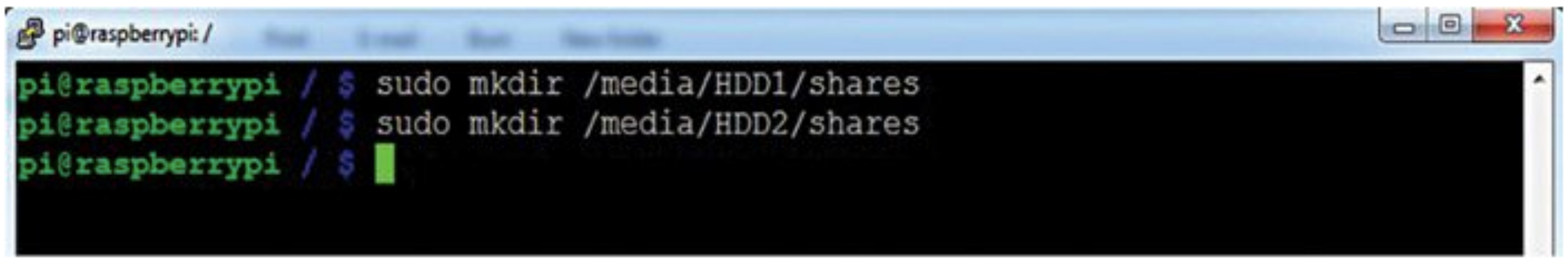
$ sudo mkdir /media/HDD1
$ sudo mkdir /media/HDD2

you can mount the drives to each location by executing the following commands:

$ sudo mount -t auto /dev/sda1
/media/HDD1
$ sudo mount -t auto /dev/sdb1
/media/HDD2

'sda1' and 'sdb1'are names of the external drives; replace them with the ones you get as shown in Fig. 2.

Now we have both the drives mounted; we just need to create a folder in both directories to hold all the shared folders. refer Fig. 4 and write the commands:

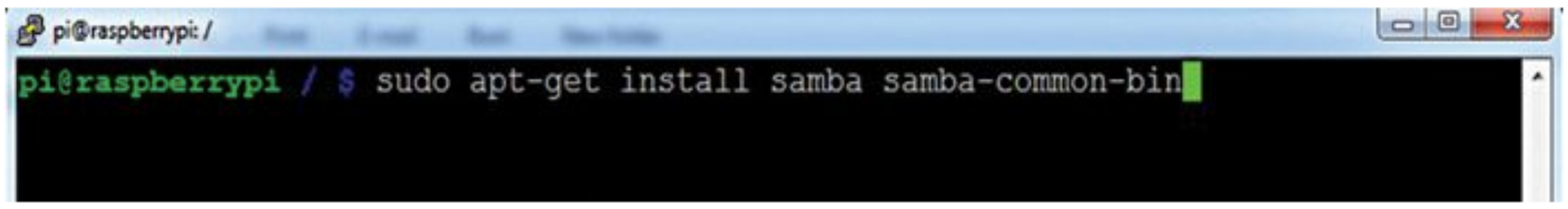


```
pi@raspberrypi / $ sudo mkdir /media/HDD1/shares
pi@raspberrypi / $ sudo mkdir /media/HDD2/shares
pi@raspberrypi / $
```

Fig. 4: Create the shared folder

$ sudo mkdir /media/HDD1/shares
$ sudo mkdir /media/HDD2/shares

Setting up the network



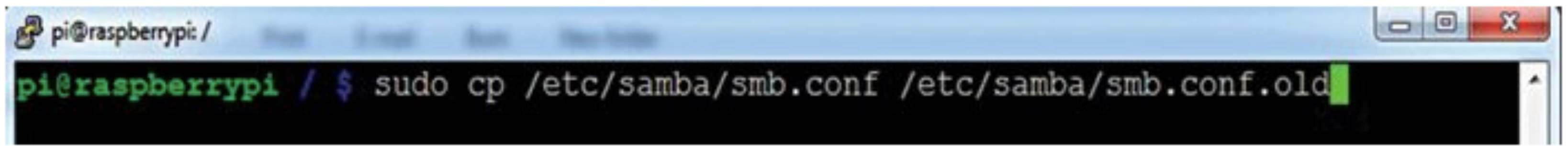
```
pi@raspberrypi / $ sudo apt-get install samba samba-common-bin
```

Fig. 5: Installing samba

Install samba for accessing the storage from anywhere on the network using the command below as shown in Fig. 5:

$ sudo apt-get install samba
samba-common-bin

Once installation is complete, we need to make some changes in samba's configuration file. But before that, create back up of the configuration file (in case we make any mistake) by executing the command below as shown in Fig. 6:



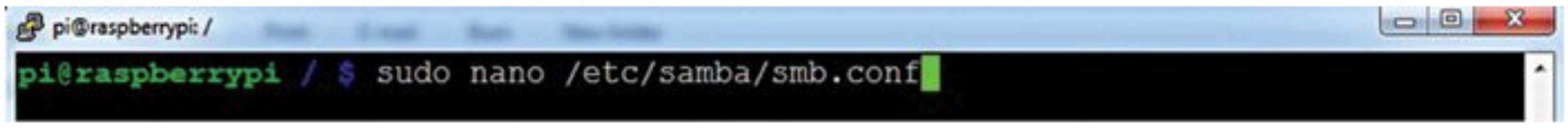
```
pi@raspberrypi / $ sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.old
```

Fig. 6: Backup of the configuration file

$ sudo cp /etc/samba/smb.conf /etc/
samba/smb.conf.old

This will create a backup of the configuration file with the name 'smb.conf.old' in the same directory.

Now to edit the samba config file, open it in nano editor by executing the command below as shown in Fig. 7:



```
pi@raspberrypi / $ sudo nano /etc/samba/smb.conf
```

Fig. 7: Edit the configuration file using nano editor

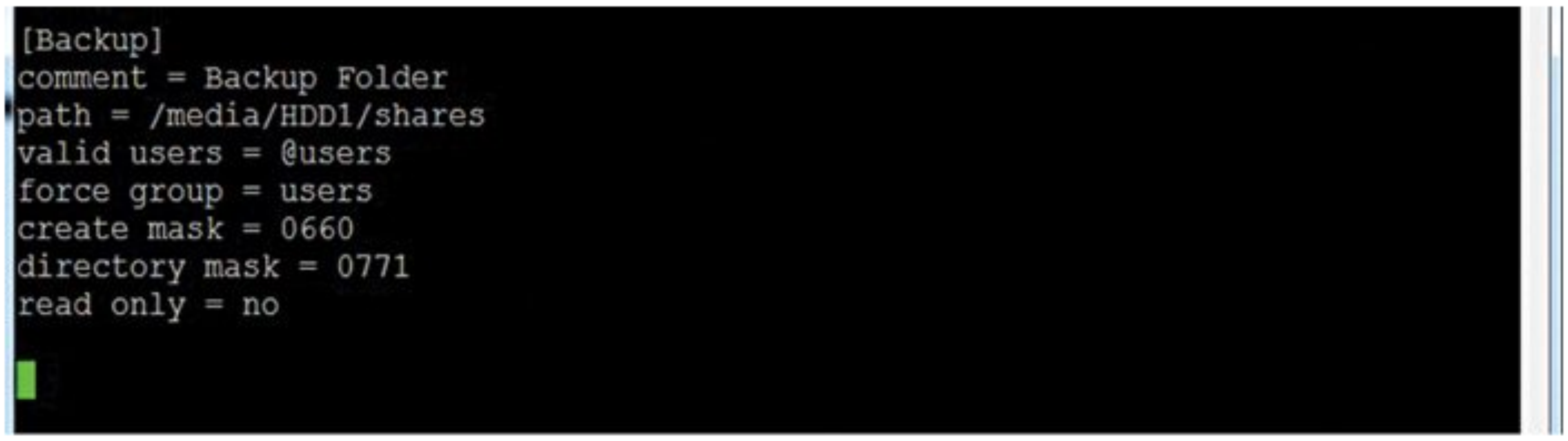
$ sudo nano /etc/samba/smb.conf

The first entry in this file is workgroup identifier; by default it is 'workgroup = WORKGROUP.' If you are using a different name for your home workgroup, go ahead and change that now; otherwise leave it as the default.

Under authentication section, add following line or remove the '#' symbol if it is already there. otherwise, anyone with network access will be able to access the shared data. This will enable username and password verification.

security = user

Now scroll down to the bottom of the file and add following commands as shown in Fig. 8:



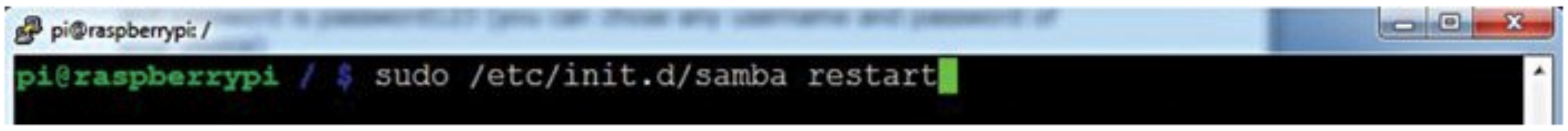
```
[Backup]
comment = Backup Folder
path = /media/HDD1/shares
valid users = @users
force group = users
create mask = 0660
directory mask = 0771
read only = no
```

Fig. 8: Adding the lines to configuration file

[Backup]
comment = Backup Folder
path = /media/HDD1/shares
valid users = @users
force group = users
create mask = 0660
directory mask = 0771
read only = no

After adding this, press 'CTRL+O' to save the file and 'CTRL+X' to exit.

Restart samba as shown in Fig. 9 with the command below:



```
pi@raspberrypi / $ sudo /etc/init.d/samba restart
```

Fig. 9: Restart samba

$ sudo /etc/init.d/samba restart

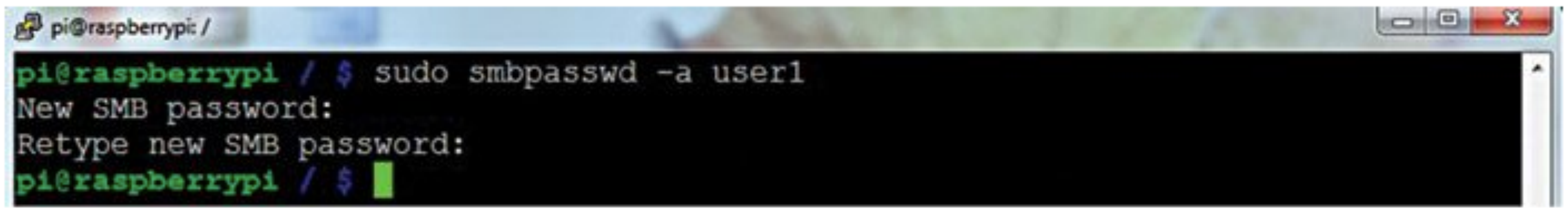
Add in a user who can access Raspberry Pi's samba with the commands below. In this case, user is 'user1' and password is 'password123.' You can choose any username and password.

$ sudo useradd
user1 –m -G users
$ sudo passwd
password123

Reconfirm the password when prompted. After confirming the password, it is time to add 'user1' as a legitimate samba user. Enter the following command:

sudo smbpasswd
-a user1

Enter the password when prompted as shown in Fig. 10.



```
pi@raspberrypi / $ sudo smbpasswd -a user1
New SMB password:
Retype new SMB password:
pi@raspberrypi / $
```

Fig. 10: Add user1 as legitimate samba user

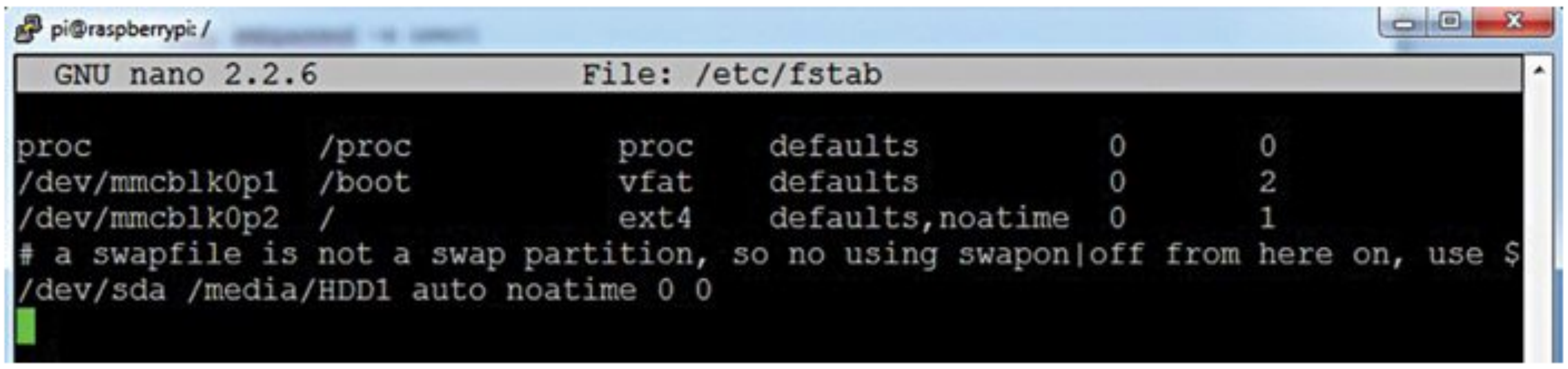
You can now use any samba-capable machine on network and test connectivity to the network using the username and password created earlier.

Configuring Pi for automatic mounting of the drives

We need to configure our Raspberry Pi such that when it restarts, it automatically mounts the external hard drives. To do so, we need to start the nano editor and make following changes:

sudo nano /etc/fstab

And add the following lines in the file as shown in Fig. 11:



```
  GNU nano 2.2.6              File: /etc/fstab

proc            /proc           proc    defaults          0       0
/dev/mmcblk0p1  /boot           vfat    defaults          0       2
/dev/mmcblk0p2  /               ext4    defaults,noatime  0       1
# a swapfile is not a swap partition, so no using swapon|off from here on, use $
/dev/sda /media/HDD1 auto noatime 0 0
```

Fig. 11: Configuration for automatic mounting of drives

/dev/sda1 /media/HDD1 auto noatime

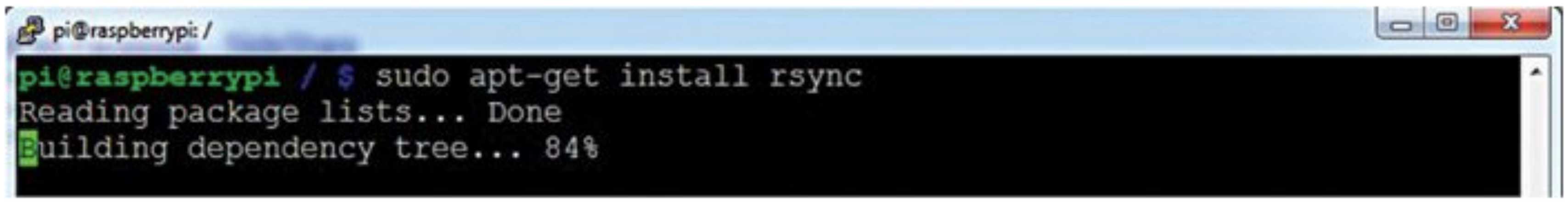
0 0
/dev/sdb2 /media/HDD2 auto noatime
0 0

After adding these lines, press ‘CTRL+O’ to save the file and ‘CTRL+X’ to exit.

Configuring Raspberry Pi for data redundancy

To configure Raspberry Pi to perform data mirroring from the share folder of the primary disk to the share folder of the secondary disk, two tools have to be used —‘cron’ and ‘rsync.’

Install rsync using the command:



```
pi@raspberrypi:/
pi@raspberrypi / $ sudo apt-get install rsync
Reading package lists... Done
Building dependency tree... 84%
```

Fig. 12: Installing the rsync

$ sudo apt-get install rsync

Once rsync is installed, set up cron to automate the process of copying files from HDD1 to HDD2. Run the command:

```
pi@raspberrypi:/
pi@raspberrypi / $ crontab -e
```

Fig. 13: For automatic process of copying files

$ crontab -e

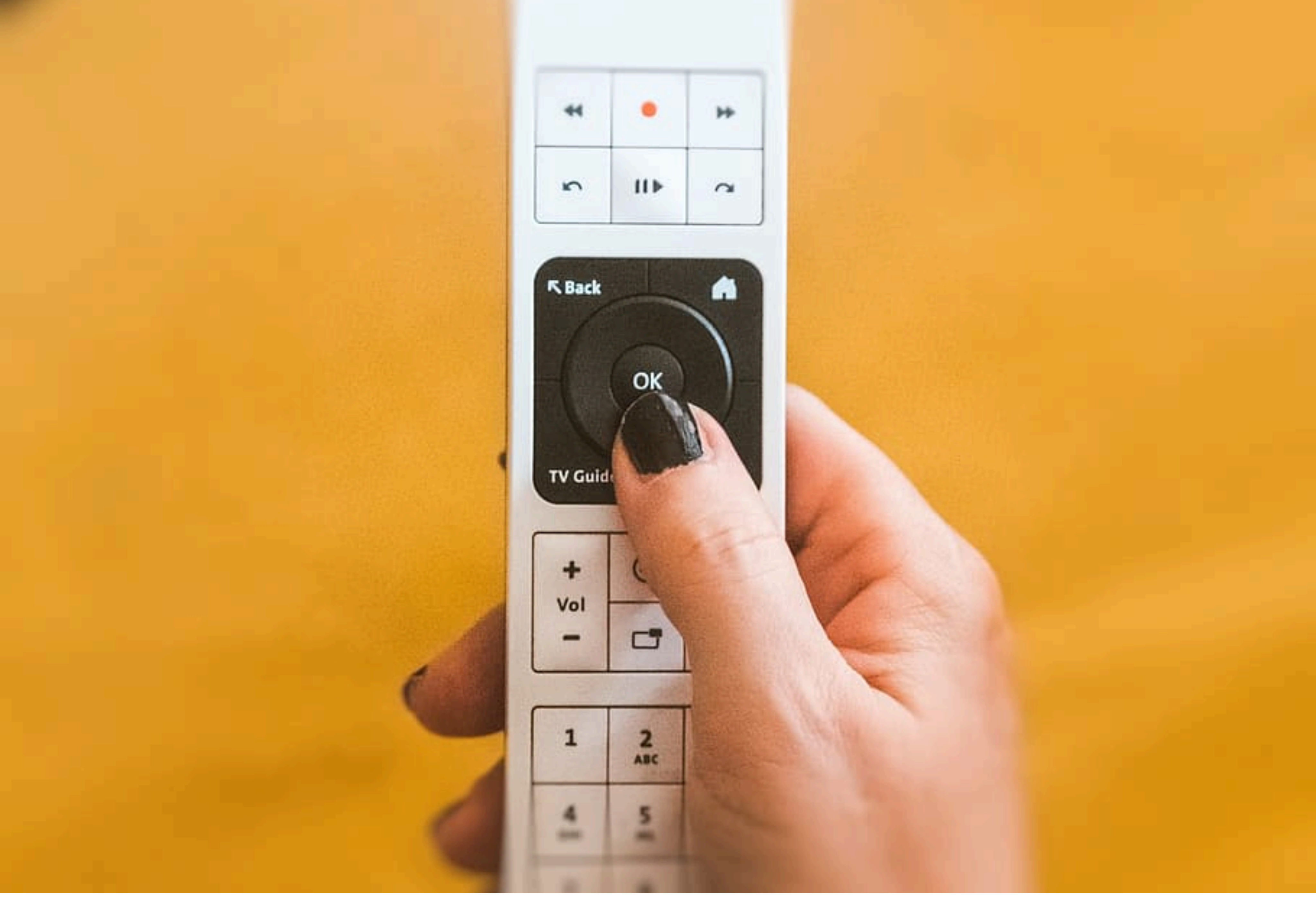
This will open up cron scheduling table. Add following line at the end of the file:

*0 5 * * * rsync -av ––delete /media/*

HDD1/shares /media/HDD2/shares/

In section 0 5 * * *, first two digits define time and the three stars are for place of year, month and day as wild card entries.

Thus each day at 5:00 am, rsync will compare the share folders of the primary and secondary drives and copy all new files to secondary drive. It will also delete those from the secondary drive which are no longer present in the primary drive. Now save and exit the editor.



42

REMOTE CONTROL FOR HOME APPLIANCES

BY S. MOHAN

Connect this circuit to any of your home appliances (lamp, fan, radio, etc) to make the appliance turn on/off from a TV, VCD or DVD remote control. The circuit can be activated from up to 10 metres.

The influx of the Internet of things (IoT) devices has created a huge demand for connecting everything to everything. In the same race, this remote control for home appliances lets you connect your regular everyday appliances to be controlled by a remote. All you have to do is connect this circuit to any of your home appliances (lamp, fan, radio, etc) and you are good to go. The appliance can now be controlled by a remote control working at the designated frequency. The circuit can be activated from up to 10 metres.

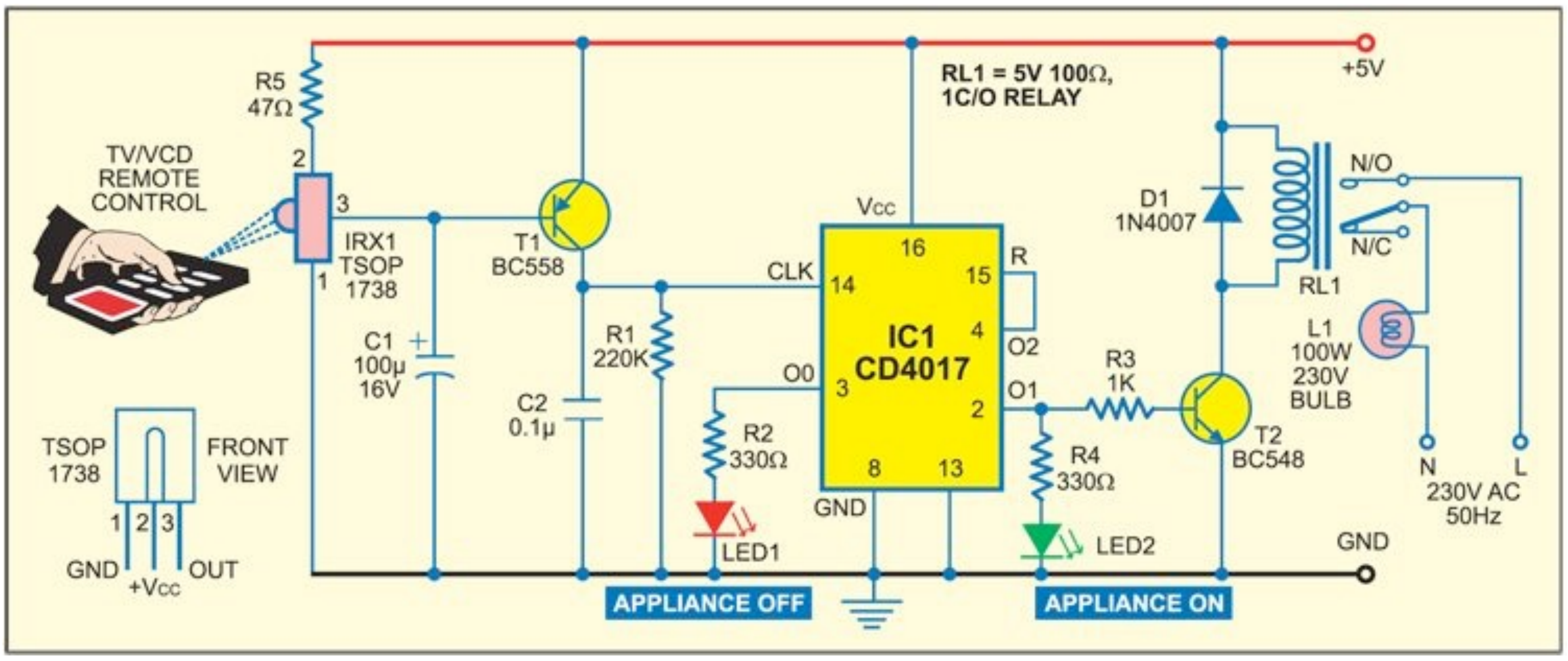
Remote control for home appliances: Working Basics

The 38kHz infrared (IR) rays generated by the remote control are received by IR receiver module TSOP1738 of the circuit. Pin 1 of TSOP1738 is connected to ground, pin 2 is connected to the power supply through resistor R5 and the output is taken from pin 3. The output signal is amplified by transistor T1 (BC558).

The amplified signal is fed to clock pin 14 of decade counter IC CD4017 (IC1). Pin 8 of IC1 is grounded, pin 16 is connected to Vcc and pin 3 is connected to LED1 (red), which glows to indicate that the appliance is 'off.'

The output of IC1 is taken from its pin 2. LED2 (green) connected to pin 2 is used to indicate the 'on' state of the appliance. Transistor T2 (BC548) connected to pin 2 of IC1 drives relay RL1. Diode 1N4007 (D1) acts as a freewheeling diode. The appliance to be controlled is connected between the pole of the relay and neutral terminal of mains. It gets connected to live terminal of AC mains via normally opened (N/O) contact when the relay energises.

Circuit Diagram



Demo Video of Remote Control for Home Appliances (Credit: Sriram Karthik)
https://youtu.be/RhqKbxGOWFY

Hope this remote control for home appliances made your life a little easier. Another project titled make IR remote should also be helpful.

If you have any projects that you would like to share with the electronics engineering community please mention it in the comments below.

43

AUTOMATIC WATER TAP WITHOUT USING MICROCONTROLLER

BY ASHWINI KUMAR SINHA

Water scarcity is one of the biggest issues plaguing the world today. To prevent this from becoming any worse, we need to focus on water conservation.

Water is the most precious resource on earth, and it is our responsibility to conserve it. But knowingly or unknowingly, we tend to waste a lot of water every day. One of the most obvious water-wasting habits is forgetting to turn off the tap after using it or leaving the water running when brushing our teeth, shaving or doing the dishes.

To help reduce water wastage from taps, we bring to you a smart solution. Today in this DIY project, we will be making a smart water tap that automatically turns off when you are not using it. It will automatically turn on when it detects hand or glass near it and turn off after the container is filled.

This automatic tap can be used in various sectors like industrial automation and even in water vending machine.

We are going to use two I.R proximity sensors in our prototype. While one sensor detects the object or hand near it to turn on/off the water flow, another sensor is mounted on top of the tap to detect the water level. When this sensor detects that the container is filled with water up to the top, it immediately cuts the flow of water.

Components we need

- 2 Channel Relay Boards
- 2 I.R sensor Modules
- Solenoid valve
- Some wires
- A water tap
- Water pipe
- 5V DC battery
- 12 to 24 V A.C to D.C Power adapter (according to the requirement of your solenoid valve)

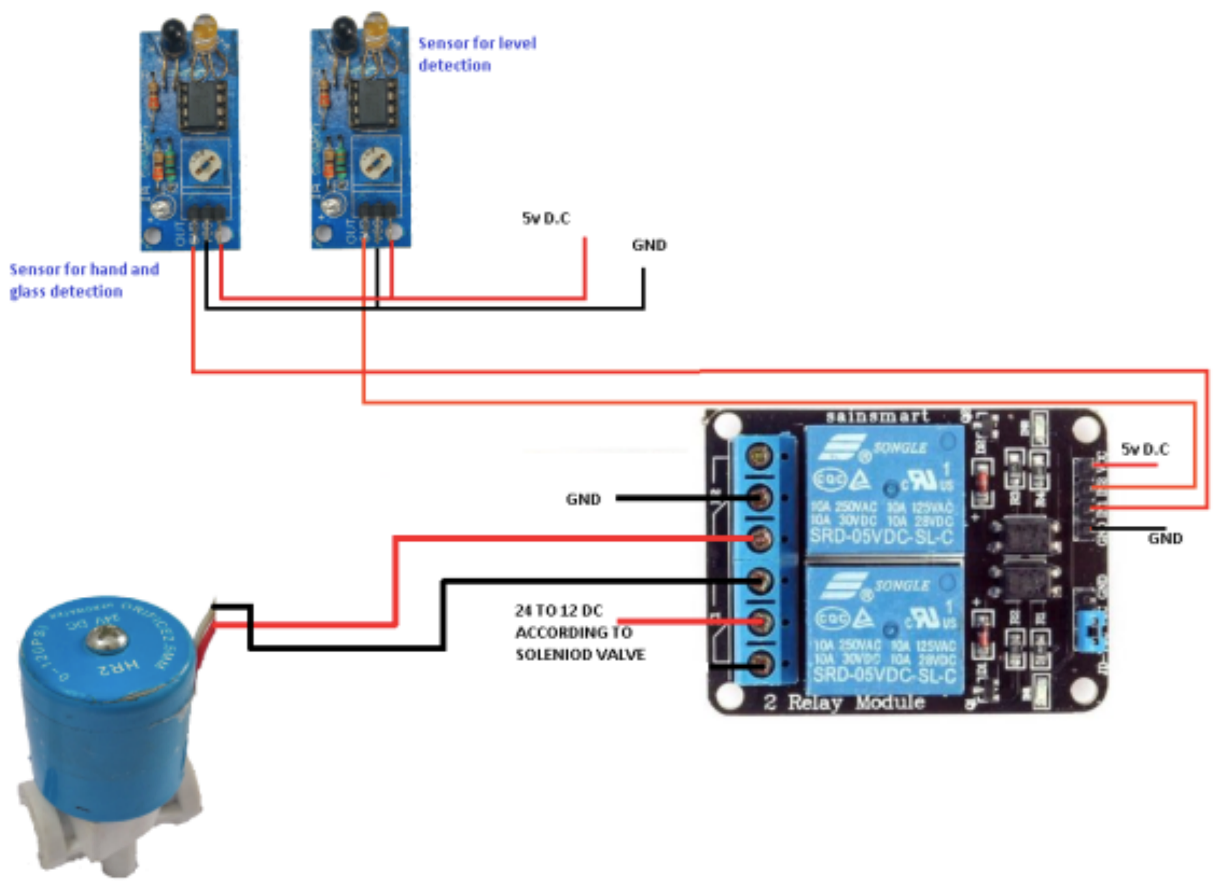

Sensor for level detection
Sensor for hand and glass detection
5v D.C
GND
5v D.C
GND
GND
24 TO 12 DC ACCORDING TO SOLENIOD VALVE
2 Relay Module

Fig 1. Circuit Diagram for Smart Water Tap

Connection

First, we need to connect the ground wires of the sensors and relay module to the negative terminal of a 5V battery. After that join the VCC wires of the relay module and sensors to the positive terminal of the battery. Then connect the output of the first sensor to relay input pin 1 and second sensor output pin to relay module input pin 2.

Sensor to relay input pin 1 and second sensor output pin to relay module input pin 2.

Then connect the solenoid valve one wire to N.C (Normally Close) part of relay 1 and another wire to No (Normally Open) part of relay 2. After this connect the common 1 of relay 1 to negative 24 V DC power and common pin of relay 2 to negative of 24 V DC power supply. Refer Fig. 1 for the connection of components.

Working and Testing of Smart Water Tap System

Our circuit depends on two sensors for its operation. While one sensor detects the object/hand near it, the other sensor detects the level of water filled in the container. When the first sensor detects an object, it gives signal to relay 1 that results in the movement of plunger of relay module towards Normally Open and the valve gets connected to the "Normally Open 1" wire of the relay module. After the circuit is completed, water starts flowing through the valve. When the sensor 2 detects that the glass/container is filled up to the top, it gives signal to the second relay input and the plunger of relay move to disconnect the valve circuit, automatically turning off the water flow.This system works more accurately when you use the colored soft drink or with any juice.With water due to low reflection the flow of water will stop when the water level touch the I.R sensor.

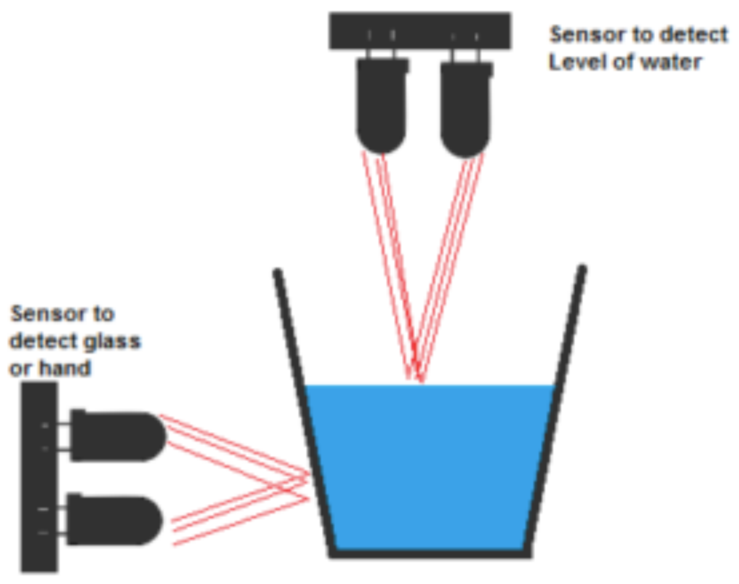
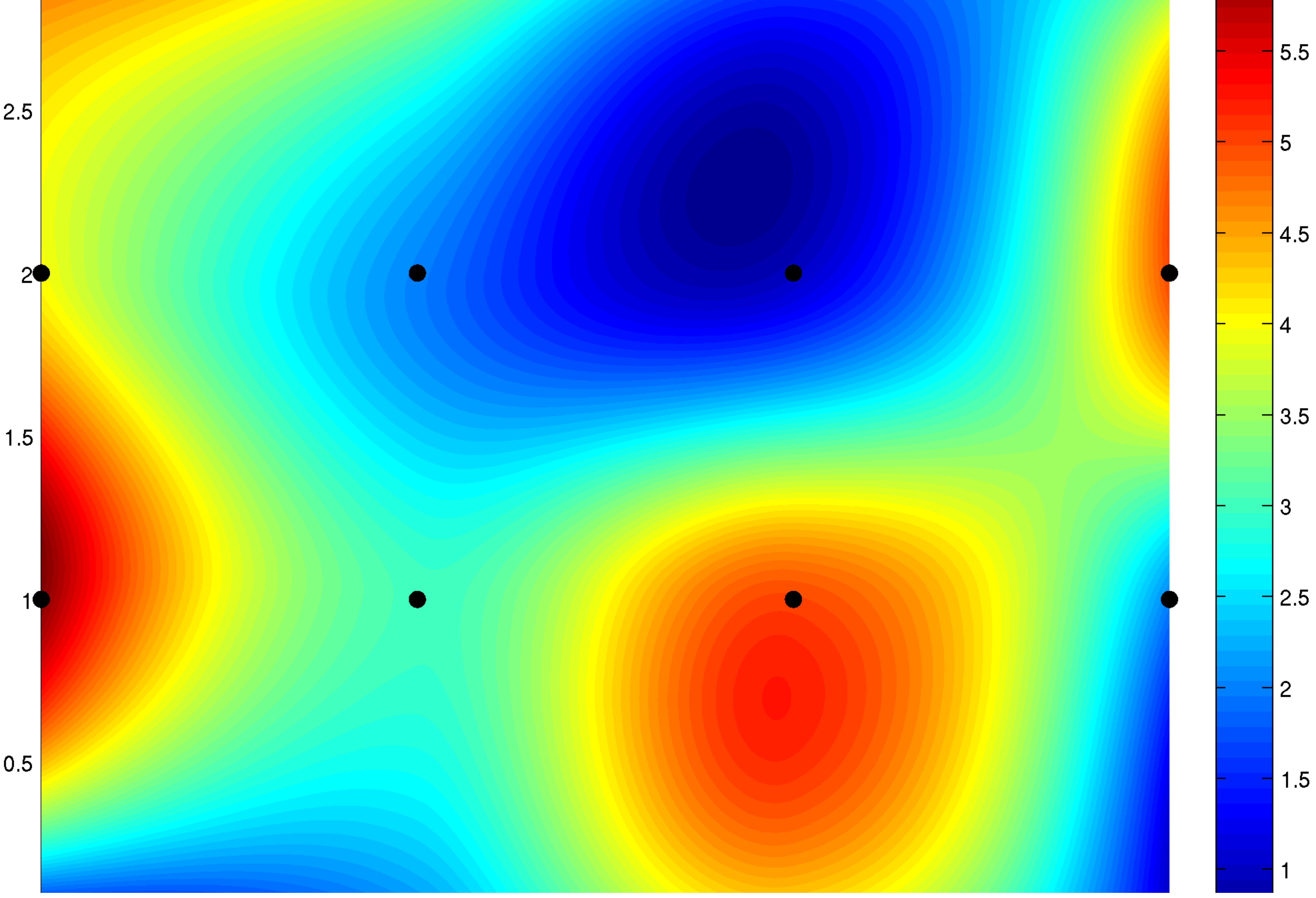


Fig 2. Working of Smart Water Tap



44

BASIC IMAGE PROCESSING USING MATLAB

BY ISMAIL TAIBANI

In this article, the author describes basic image processing using MATLAB software.

MATLAB is a high-performance language for technical computing with powerful commands and syntax. It is used for many purposes like Maths and computation, data analysis, algorithm development, modelling stimulation and prototyping. Edge detection, noise and image histogram modelling are some important and basic topics in image processing.

Image processing using MATLAB

Edge detection

An image is nothing but mapping of intensity of the light reflecting from a scene captured from a camera, and edges are the discontinuity of the scene intensity function. We can detect these edges using MATLAB commands. There are many methods for edge detection such as Robert's operator, Prewitt operator, Sobel operator, Canny edge detector and so on. Fig. 1 shows edge detection using these operators on cameraman.tif standard image available in MATLAB.

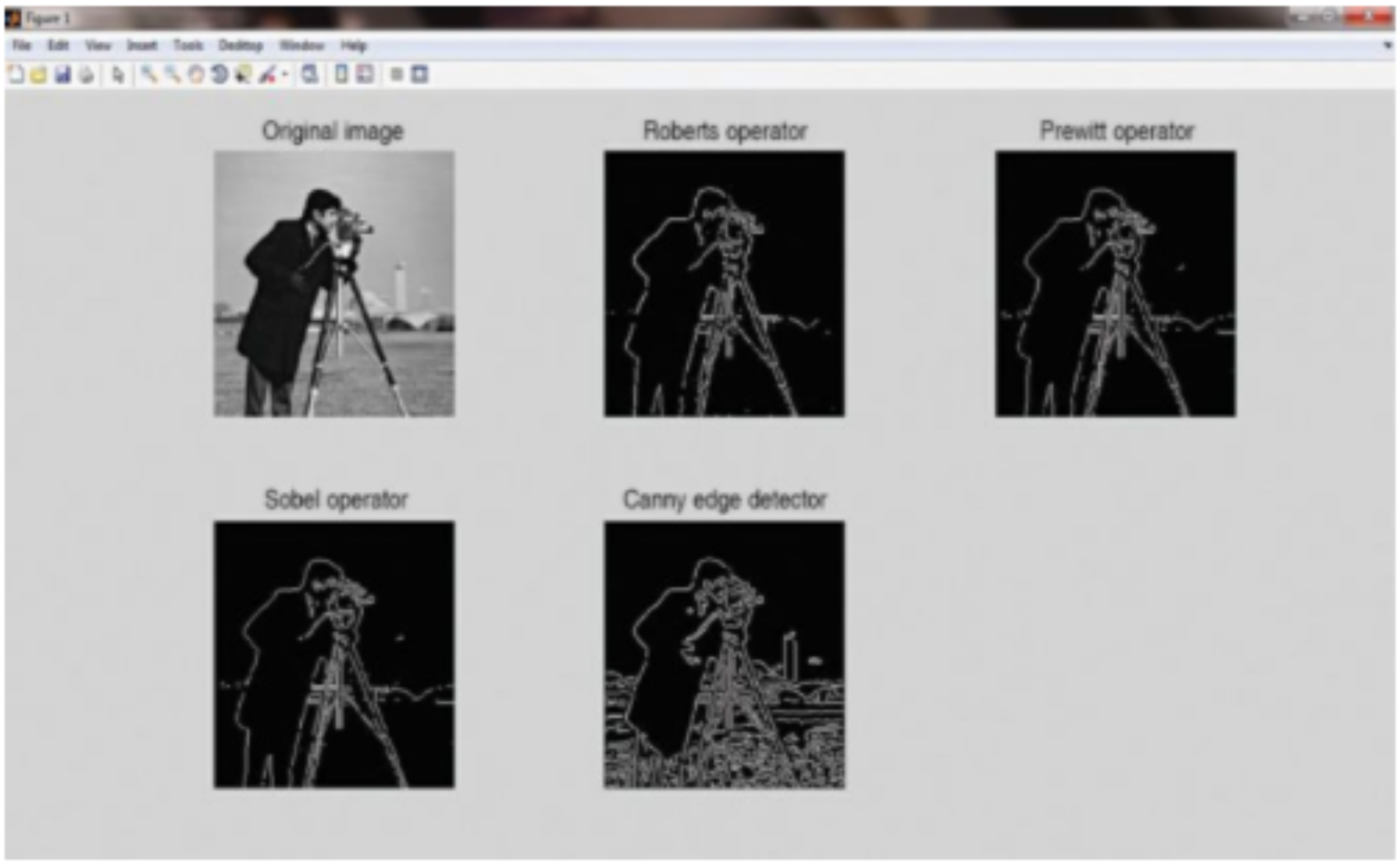


Fig. 1: Edge detection using various operators

Noise

Noise in any system is unwanted. In image processing, noise in a digital image arises during image acquisition and also during transmission. Different types of noise include speckle, Gaussian, salt-and-pepper and more. The fun part is, we can use these types of noise as special effects in an image using MATLAB.

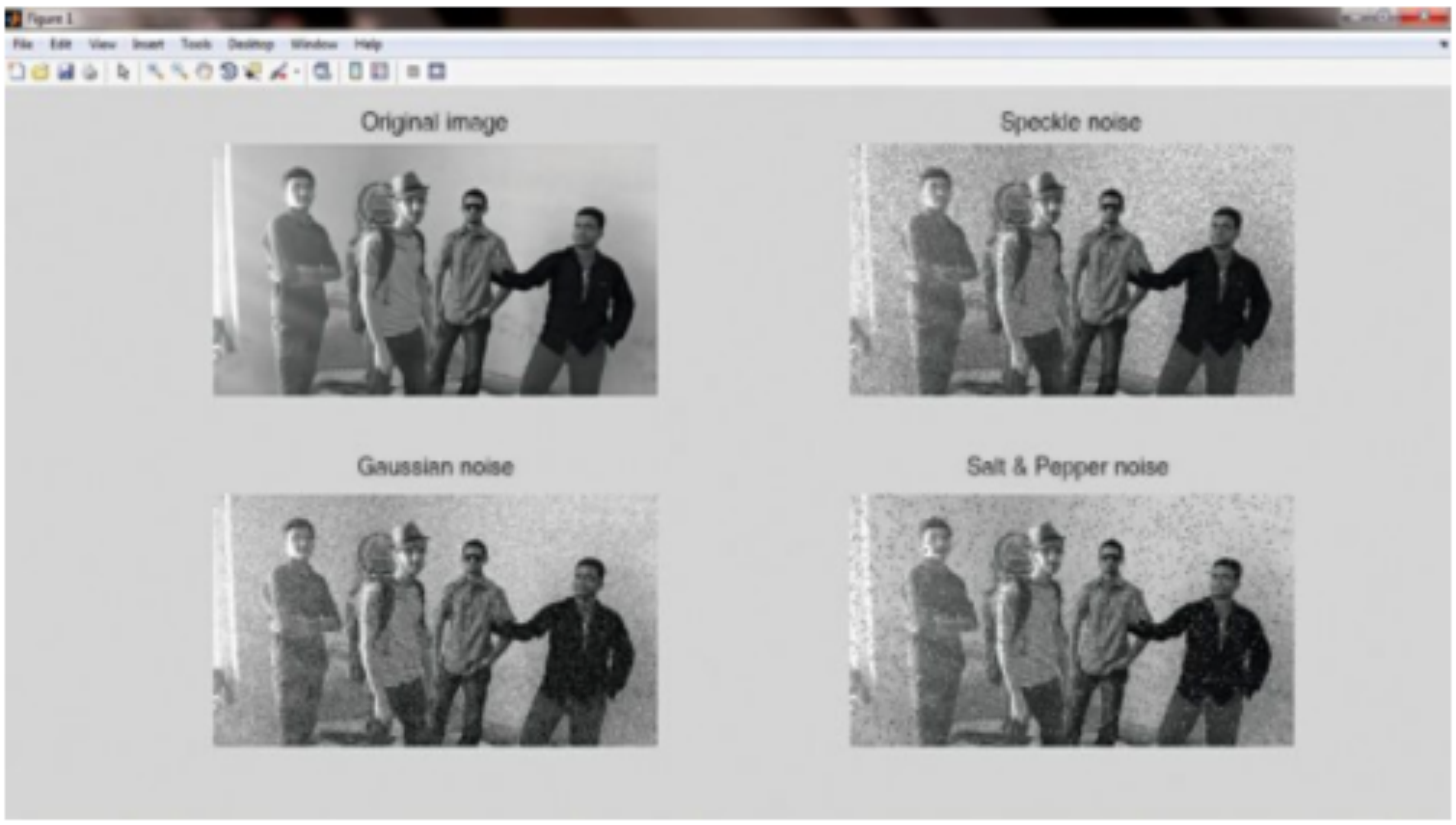


Fig. 2: Special effects in an image using different types of noise

Fig. 2 shows the results of different types of noise added to an image. In this image, RGB-to-gray conversion is done first and then different types of noise are added in the image through the program. All operations are included in MATLAB program.

Histogram modelling

A histogram of an image provides a vast description about an image. It represents the occurrence of various gray levels relative to the frequencies. In this program, we plot the histogram of the original

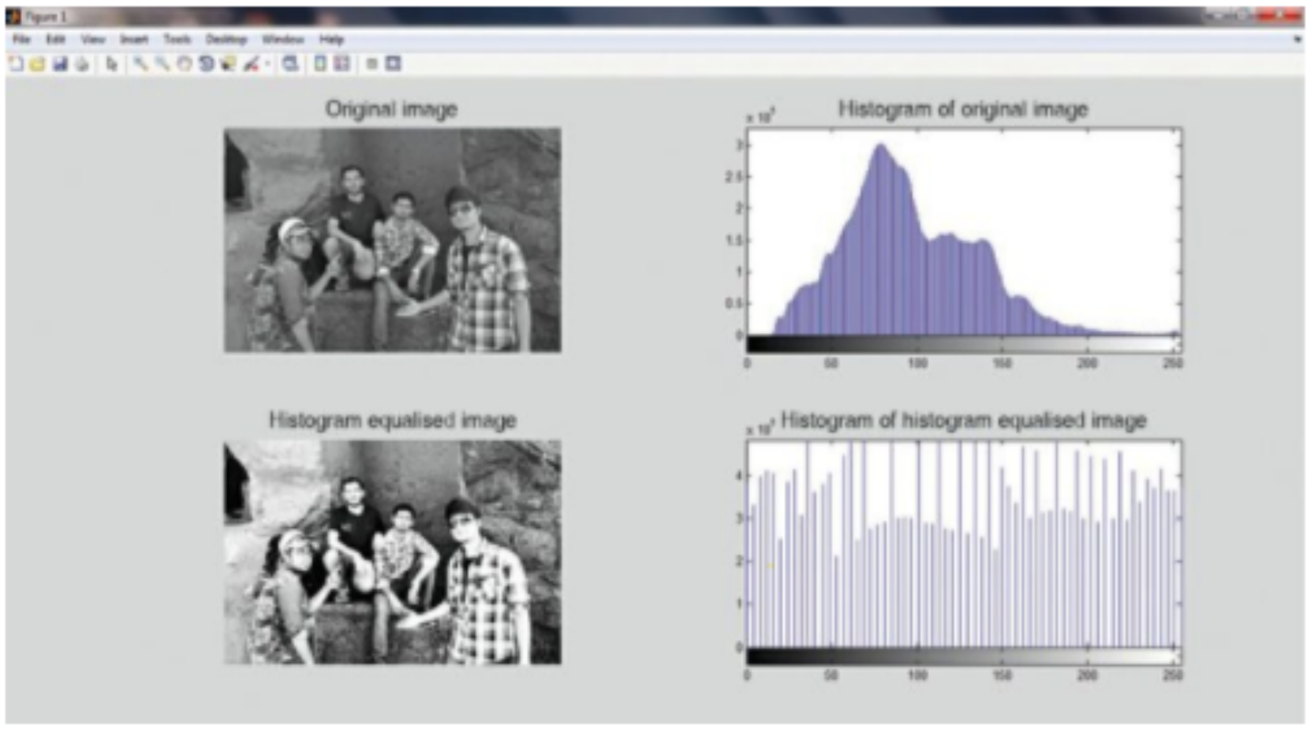


Figure 1
File Edit View Insert Tools Desktop Window Help
Original image
Histogram of original image
Histogram equalised image
Histogram of histogram equalised image
0
50
100
150
200
250

Fig. 3: Histogram modelling

image and of the histogram-equalised image.

Testing

Running the program is straightforward. There are three .m files, one each for edge detection, noise effects and histogram. Two image files (.jpeg) are also included along with these .m files in the same folder. Launch MATLAB R2013a from your desktop and open an .m file from C:\Users\SONY\Desktop folder to run the program.

Image processing is a diverse and the most useful field of science, and this article gives an overview of image processing using MATLAB. There are many more topics that are useful and can be applied using MATLAB or OpenCV library such as erosion, dilation, thresholding, smoothing, degradation and restoration, segmentation part like point processing, line processing and edge detection (covered here) of images.

Download source code: click here

45

OBJECT FINDER ROBOT

BY ASHWINI KUMAR SINHA

The Object Finder Robot uses OpenCV and computer vision along with a pre-trained model of TensorFlow running on the Raspberry Pi for detecting and classifying an object. In this robot, we have connected a servo motor that is mounted with a camera on top of it for capturing an overall continuous video of the surroundings. Using the TensorFlow module, it will try to detect the object.

Bill of Materials

Component Name	Quantity	Description	Cost Approx. In INR
Raspberry Pi 3/4,Zero	1	Any Model Of Raspberry Pi	1500
Camera	1	Raspberry Pi Camera	300
Servo Motor	1	Servo Motor	300
Total Cost			**5430**

Coding

First, we need to install the following libraries.

- Guizero
- dlib
- Tensorflow
- Opencv
- Matplot
- scipi

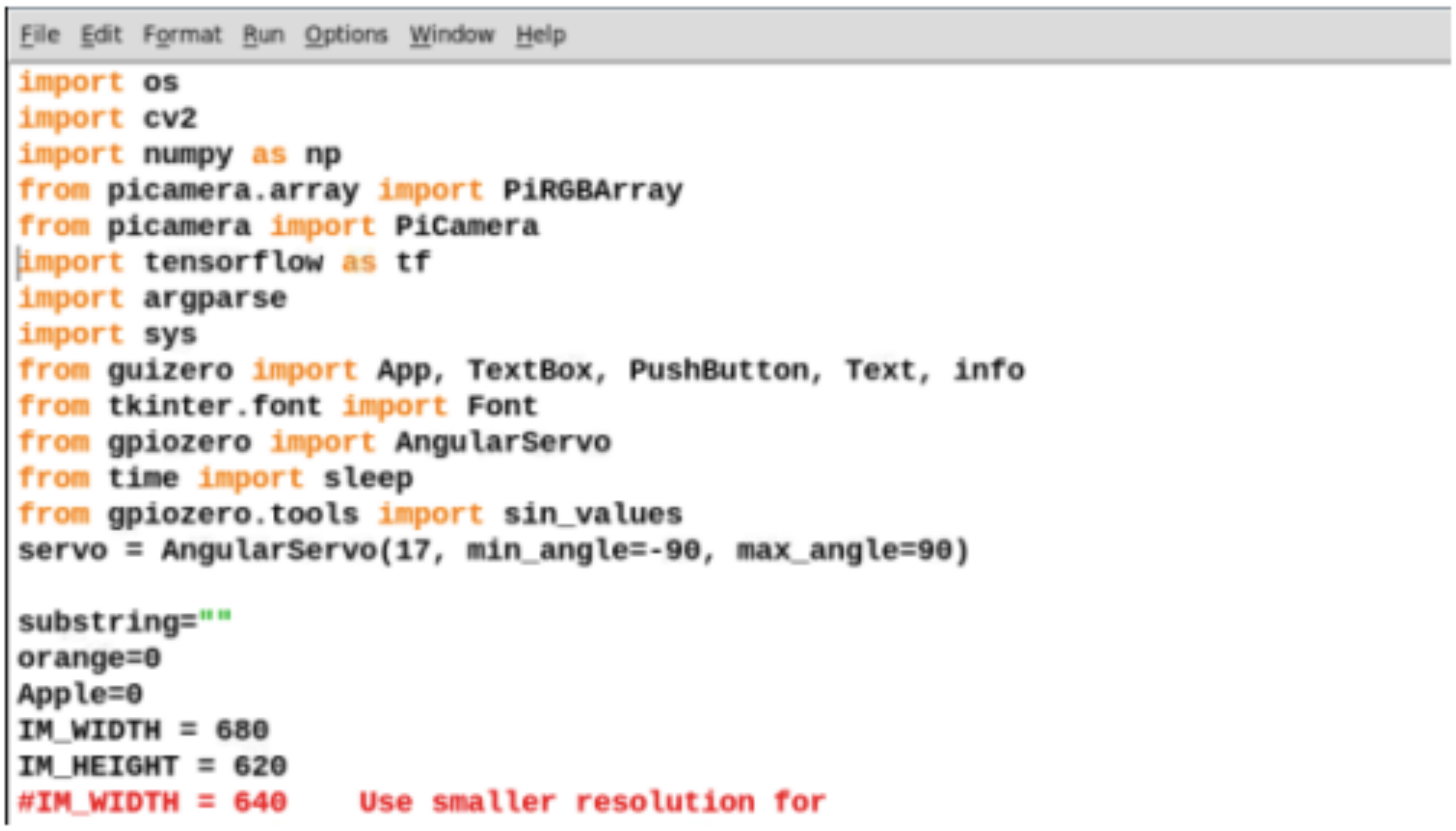
Use the following syntax to install the libraries

```
sudo pip3 install library name
sudo  apt-get update
sudo apt-get upgrade
sudo nano /etc/dphys-swapfile
Then change the line CONF_SWAPSIZE=100 to  CONF_SWAPSIZE=1024
sudo /etc/init.d/dphys-swapfile stop
sudo /etc/init.d/dphys-swapfile start
sudo pip3 install opencv
sudo pip3 install numpy
wget https://bootstrap.pypa.io/get-pip.py
pip3 install dlib
pip3 install tensorflow
```

After the installation, you can now proceed with the cloning of TF modules, examples and files using the following command:

git clone https://github.com/tensorflow/tensorflow.git

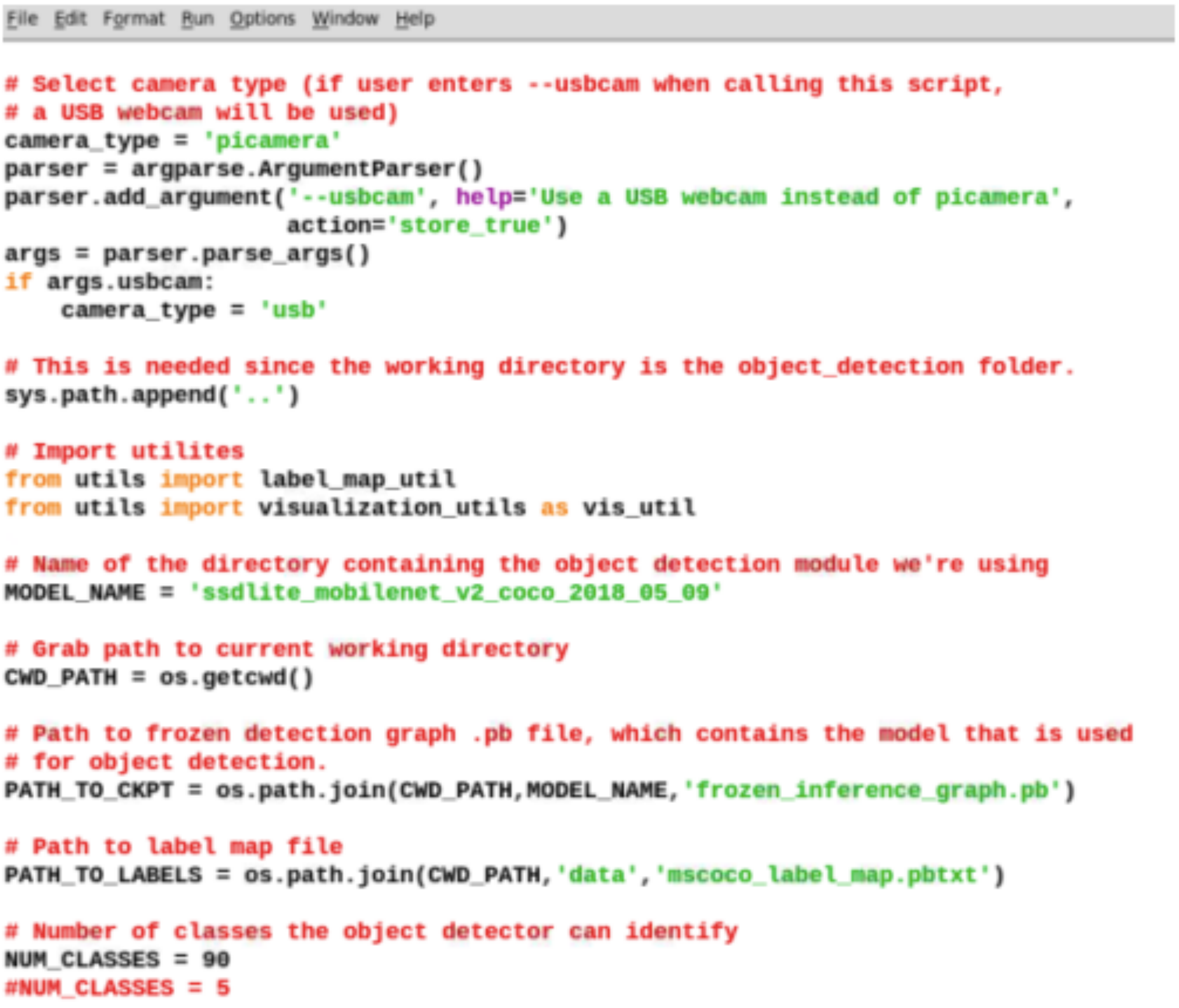
After installing the required libraries, we can now start our coding. Open the Python IDE and create a new file in the tensor flow object detection folder. Then, import the installed libraries in the code and set the pin for the servo motor.



```
import os
import cv2
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
import tensorflow as tf
import argparse
import sys
from guizero import App, TextBox, PushButton, Text, info
from tkinter.font import Font
from gpiozero import AngularServo
from time import sleep
from gpiozero.tools import sin_values
servo = AngularServo(17, min_angle=-90, max_angle=90)

substring=""
orange=0
Apple=0
IM_WIDTH = 680
IM_HEIGHT = 620
#IM_WIDTH = 640    Use smaller resolution for
```

Fig 1

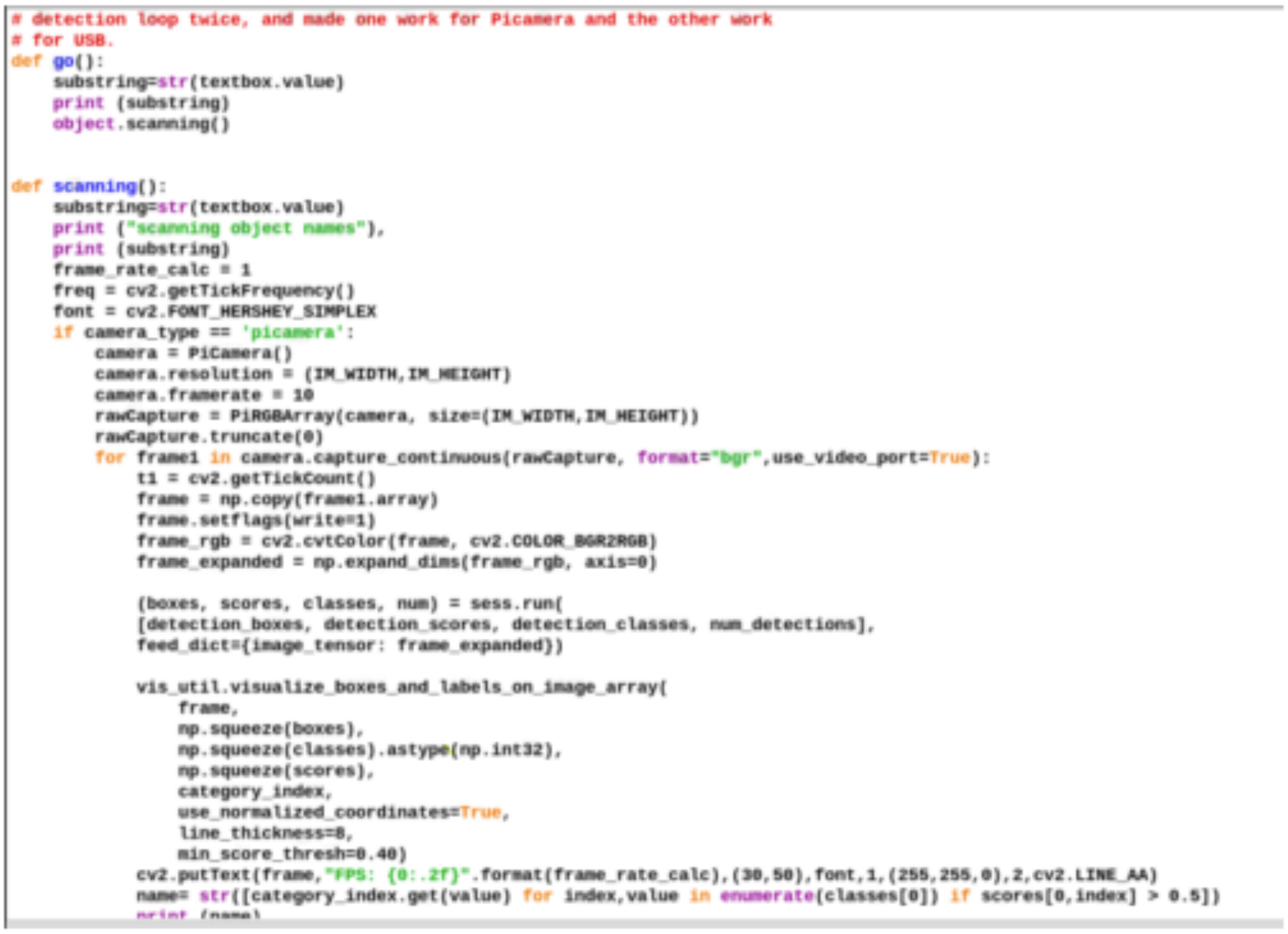
Now we will write a code that will make the GUI ask for the name of the object that you want to find. The GUI opens in a new window in which the name of the object to be found has to be entered. After that, we will create a code that converts the input into a string and saves that as an object to be detected. Next, we will set the path for the object detection model.



```
# Select camera type (if user enters --usbcam when calling this script,
# a USB webcam will be used)
camera_type = 'picamera'
parser = argparse.ArgumentParser()
parser.add_argument('--usbcam', help='Use a USB webcam instead of picamera',
                    action='store_true')
args = parser.parse_args()
if args.usbcam:
    camera_type = 'usb'

# This is needed since the working directory is the object_detection folder.
sys.path.append('..')

# Import utilites
from utils import label_map_util
from utils import visualization_utils as vis_util

# Name of the directory containing the object detection module we're using
MODEL_NAME = 'ssdlite_mobilenet_v2_coco_2018_05_09'

# Grab path to current working directory
CWD_PATH = os.getcwd()

# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,'data','mscoco_label_map.pbtxt')

# Number of classes the object detector can identify
NUM_CLASSES = 90
#NUM_CLASSES = 5
```

Fig 2

Next, we will create a function scanning feature that captures the video from the camera, cuts it frame-by-frame, tries to recognise the object and then labels that with a name to give the desired result. That result will then be converted in the form of a string. So, by searching the name of the object in the detected object result, we can find out whether the scanner has detected that object.

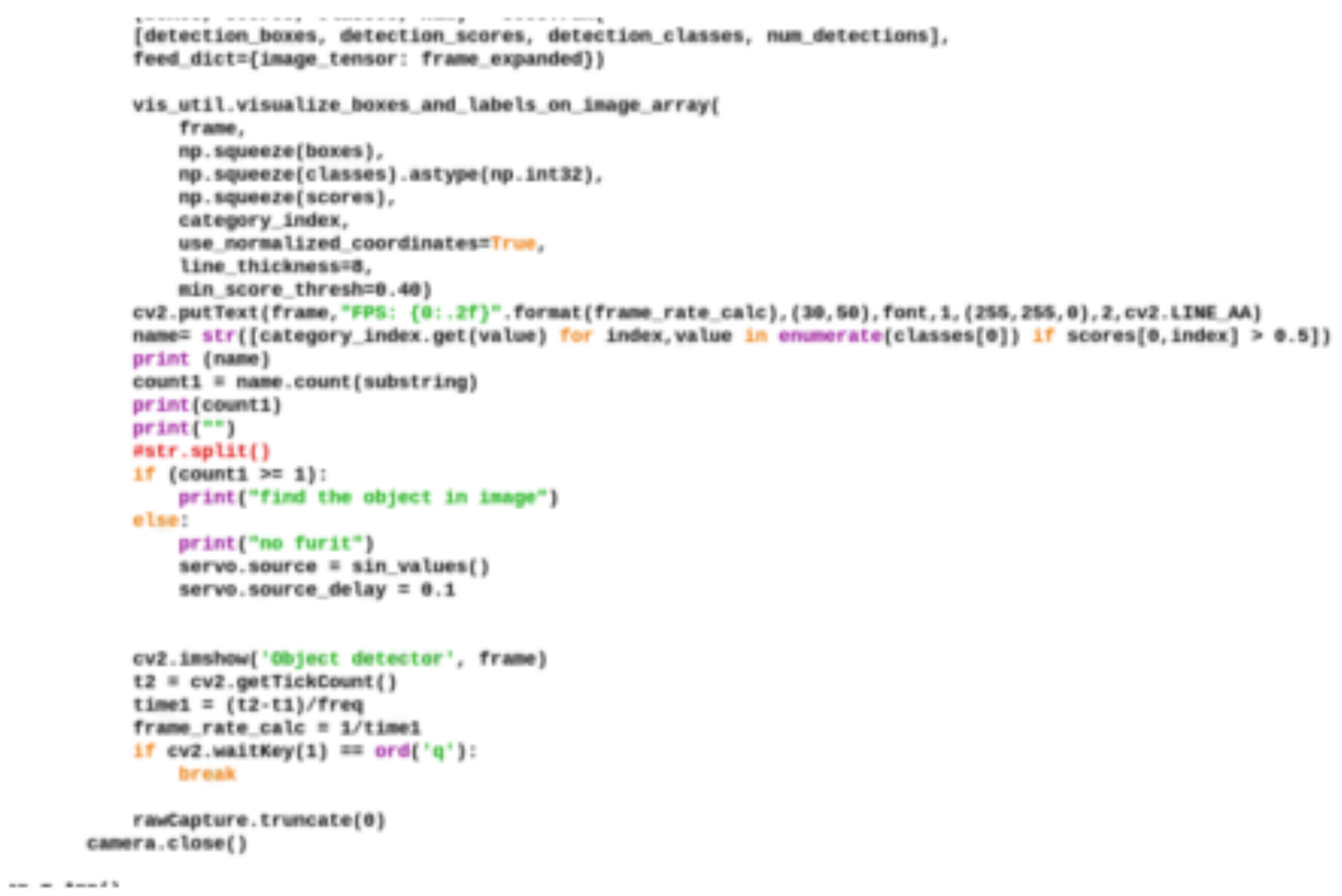


```
# detection loop twice, and made one work for Picamera and the other work
# for USB.
def go():
    substring=str(textbox.value)
    print (substring)
    object.scanning()


def scanning():
    substring=str(textbox.value)
    print ("scanning object names"),
    print (substring)
    frame_rate_calc = 1
    freq = cv2.getTickFrequency()
    font = cv2.FONT_HERSHEY_SIMPLEX
    if camera_type == 'picamera':
        camera = PiCamera()
        camera.resolution = (IM_WIDTH,IM_HEIGHT)
        camera.framerate = 10
        rawCapture = PiRGBArray(camera, size=(IM_WIDTH,IM_HEIGHT))
        rawCapture.truncate(0)
        for frame1 in camera.capture_continuous(rawCapture, format="bgr",use_video_port=True):
            t1 = cv2.getTickCount()
            frame = np.copy(frame1.array)
            frame.setflags(write=1)
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frame_expanded = np.expand_dims(frame_rgb, axis=0)

            (boxes, scores, classes, num) = sess.run(
            [detection_boxes, detection_scores, detection_classes, num_detections],
            feed_dict={image_tensor: frame_expanded})

            vis_util.visualize_boxes_and_labels_on_image_array(
                frame,
                np.squeeze(boxes),
                np.squeeze(classes).astype(np.int32),
                np.squeeze(scores),
                category_index,
                use_normalized_coordinates=True,
                line_thickness=8,
                min_score_thresh=0.40)
            cv2.putText(frame,"FPS: {0:.2f}".format(frame_rate_calc),(30,50),font,1,(255,255,0),2,cv2.LINE_AA)
            name= str([category_index.get(value) for index,value in enumerate(classes[0]) if scores[0,index] > 0.5])
            print (name)
```

Fig 3

This piece of code allows the servo to move or “sweeps the camera to 180 degrees” for finding the object.

After completing the code, connect the servo motor as shown in the circuit diagram and then mount the camera on the shaft of the servo motor.



```
        [detection_boxes, detection_scores, detection_classes, num_detections],
        feed_dict={image_tensor: frame_expanded})

        vis_util.visualize_boxes_and_labels_on_image_array(
            frame,
            np.squeeze(boxes),
            np.squeeze(classes).astype(np.int32),
            np.squeeze(scores),
            category_index,
            use_normalized_coordinates=True,
            line_thickness=8,
            min_score_thresh=0.40)
        cv2.putText(frame,"FPS: {0:.2f}".format(frame_rate_calc),(30,50),font,1,(255,255,0),2,cv2.LINE_AA)
        name= str([category_index.get(value) for index,value in enumerate(classes[0]) if scores[0,index] > 0.5])
        print (name)
        count1 = name.count(substring)
        print(count1)
        print("")
        #str.split()
        if (count1 >= 1):
            print("find the object in image")
        else:
            print("no furit")
            servo.source = sin_values()
            servo.source_delay = 0.1


        cv2.imshow('Object detector', frame)
        t2 = cv2.getTickCount()
        time1 = (t2-t1)/freq
        frame_rate_calc = 1/time1
        if cv2.waitKey(1) == ord('q'):
            break

        rawCapture.truncate(0)
    camera.close()
```

Fig 4

Connection

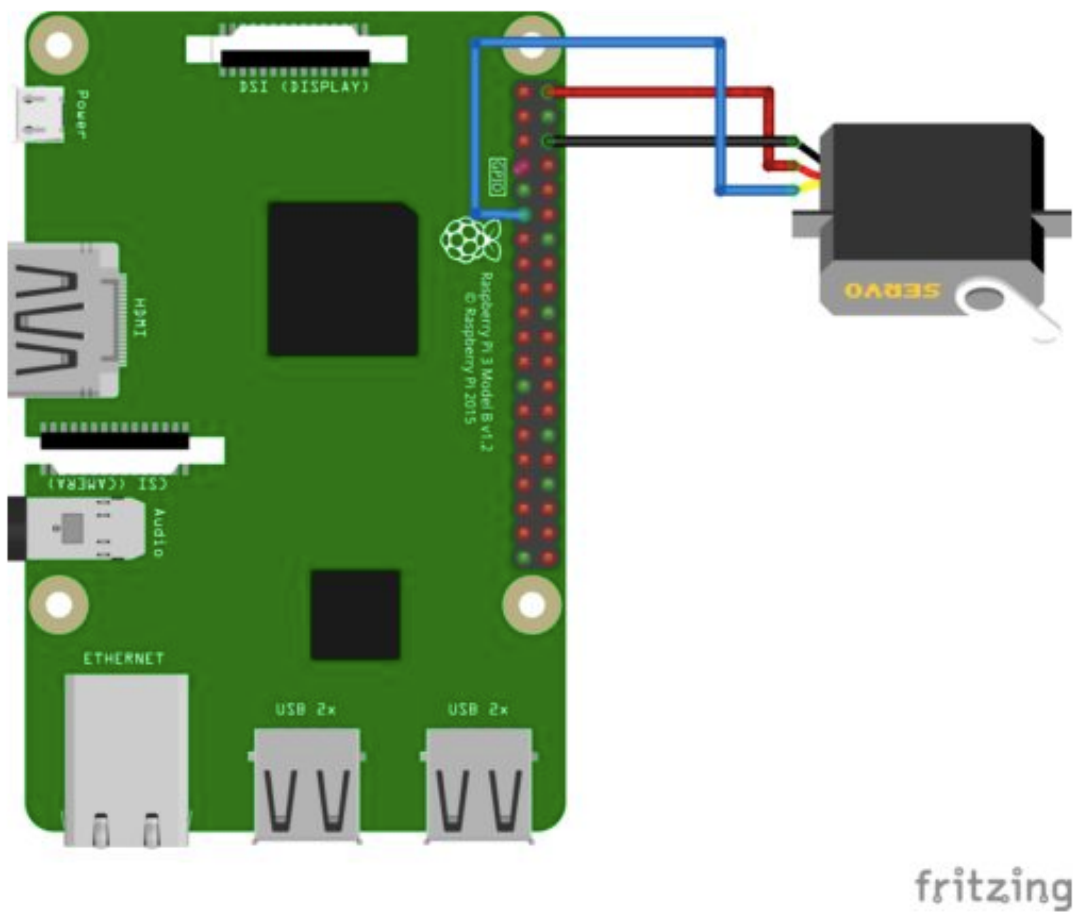


Fig 5.Connection

Testing

When you run the code, a window will appear asking you the name of the object that you want to find. Enter the name of that object in the text box and then click on find. Again a new window will appear on your device screen showing the video footage from the camera. The movement of the servo shaft will enable the camera to scan the surroundings for finding the object. When detected, the servo stops there and informs you that the object has been found.

46

ANTI-CARJACK SYSTEM

BY RAJ K. GORKHALI

Consider this: A criminal stops your car by force and you are left with no option but to surrender the car. The criminal drives away your car, but about 60 seconds later the engine stops running on its own and won't restart. Unable to use the dead vehicle, the perpetrator has to abandon it. Carjacking can be prevented using the anti-carjack system described here. The circuit automatically senses carjacking and stops the vehicle. In case the circuit is accidentally tripped off, it can be easily reset using a hidden switch. Two LEDs indicate the status of the system to the vehicle driver.

Circuit description

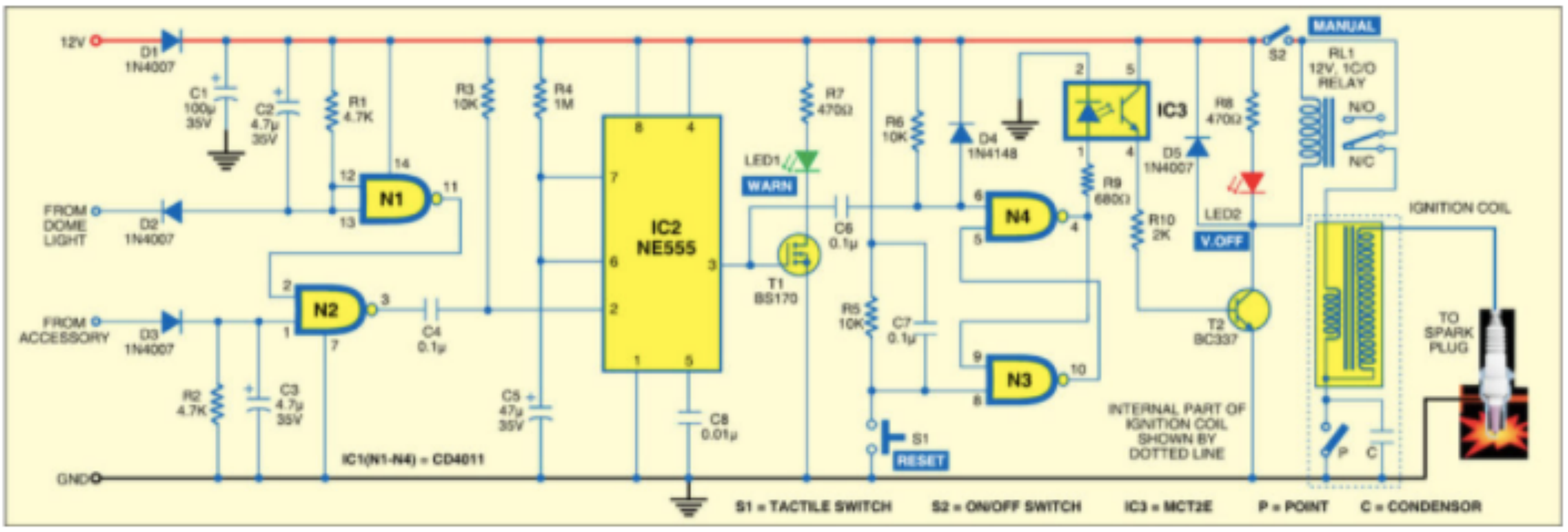


Fig. 1: Circuit of anti-carjack system

Fig. 1 shows the circuit of the anti-carjack system. It comprises quad NAND gate CD4011 (IC1), timer NE555 (IC2), optocoupler MCT2E (IC3) and a few discrete components. The circuit is powered by 12V DC from the vehicle battery. Diode D1 protects against any possible reverse voltage. The main controlling elements of the circuit are NAND gates N1 and N2 and timer NE555.

The input to gate N2 comes from a car accessory power line (such as a cigarette lighter power socket) that is energised by the ignition switch. The input to gate N1 comes from the door switch that connects the dome light. The door switches in the vehicle provide the ground return for the dome or courtesy light when any of the doors is opened.

The line connecting the door switch to the lamp is at 12V when the door is closed and at zero when any door is open.

PARTS LIST

Semiconductors:	
IC1	- CD4011 quad NAND gate
IC2	- NE555 timer
IC3	- MCT2E optocoupler
T1	- BS170 MOSFET
T2	- BC337 npn transistor
D1-D3, D5	- 1N4007 rectifier diode
D4	- 1N4148 switching diode
LED1	- 5mm green LED
LED2	- 5mm red LED
Resistors (all ¼-watt, ±5 per cent carbon):	
R1, R2	- 4.7-kilo-ohm
R3, R5, R6	- 10-kilo-ohm
R4	- 1-mega-ohm
R7, R8	- 470-ohm
R9	- 680-ohm
R10	- 2-kilo-ohm
Capacitors:	
C1	- 100µF, 35V electrolytic
C2, C3	- 4.7µF, 35V electrolytic
C4, C6, C7	- 0.1µF ceramic disk
C5	- 47µF, 35V electrolytic
C8	- 0.01µF ceramic disk
Miscellaneous:	
RL1	- 12V, 1C/O relay
S1	- Push-to-on tactile switch
	- Ignition coil

Logic signal from the dome light is fed through diode D2 to pins 12 and 13 of gate N1, which inverts it to drive pin 2 of gate N2. At the same time, the accessory power line, which is at 12V when ignition switch is turned on, connects to input pin 1 of gate N2 through diode D3.

Under normal conditions, when the ignition switch is 'on' and the doors closed, the output of gate N2 at pin 3 is high. However, when a door is opened, the output of gate N2 at pin 3 goes low. This negative-going output from gate N2 is coupled to the trigger input (pin 2) of NE555 (IC2), which is configured as a monostable multivibrator. Output pin 3 of IC2 goes high for a period determined by the combination of resistor R4 and capacitor C5.

The two remaining NAND gates (N3 and N4) of CD4011 are connected in a bistable multivibrator configuration with two input terminals, pins 8 and 6. The bistable configuration has two stable states with logic outputs at pin 4 and pin 10 always being opposite to each other. A negative transition at the input terminal of either gate N3 or N4 causes the bistable to change the state.

When the time period of NE555 (IC2) is over, its output pin 3 goes low. This negative-going signal is coupled to the input of gate N4 causing the bistable circuit to change the state. As a result, pin 4 of gate N4 goes high.

The high output of gate N4 forward-biases the internal LED of optocoupler MCT2E (IC3), driving the internal transistor into saturation. As a result, relay RL1 energises and its normally-closed (N/C) contact opens to cut the power to the ignition coil and stall the vehicle. When reset switch S1 is pressed, a negative transition is applied to pin 8 of gate N3. Pin 4 of gate N4 goes low to de-energise the relay and allow the vehicle to start.

The green LED (LED1) driven by MOSFET T1 glows to alert the vehicle owner if the timer has been activated, either by a carjack attempt or by inadvertent opening of a door when the ignition switch is 'on'. LED1 gives the driver a warning that the circuit has been triggered and a lock-out will occur in about 60 seconds. The red LED (LED2) glows whenever relay RL1 energises to stall the vehicle.

The circuit is designed to be ‘fail safe.’ The N/C contact of RL1 is used to control vehicle operation. The relay doesn’t energise unless the circuit is powered and the timer triggered.

Any component failure may cause the relay to energise. To de-energise it, open jumper JP to cut off power to the relay coil circuit so that RL1 contacts remain closed.

Construction and testing

A single-side PCB for the anti-carjack system is shown in Fig. 3 and its component layout in Fig. 4. Assemble the circuit on a PCB as it minimises time and assembly errors. Carefully assemble the components and double-check for any overlooked error. Before inserting the ICs in the PCB, properly check the supply voltage at each IC’s supply terminals. Suitable connectors are provided on the PCB for connections to accessory, dome light and battery supply. Use suitable insulated wires for these connections.

Preliminary test. You need a 12V DC source, two SPST switches and a multimeter for preliminary testing of the circuit. One SPST switch (say, switch-1) is connected between the dome light and ground and the other SPST switch (say, switch-2) is connected between the accessory point and 12V.

Check the circuit for proper connection and polarity. Also, make sure that both SPST switches are open. Now turn the power ‘on.’ If LED1 glows, allow at least 60 seconds for it to go ‘off.’ If LED2 glows, press reset switch S1 to put it off.

Close test dome switch-1 by connecting to ground. Both LEDs should go off. Open test switch-1, then close test switch-2 to simulate ignition-‘on’ condition. Both LEDs should remain ‘off.’ Close both test switches (switch-1 and switch-2), then open them. LED1 should glow. It should continue to glow for about 60 seconds. When LED1 goes off, LED2 should light up. Use multimeter to verify the resistance between the N/C contact and the pole of relay RL1. It should be infinite. Now if you press reset switch S1, LED2 should go off.

If the circuit performs as described above, the system is ready to install in your vehicle.

Installation

Before installing the anti-carjack module in the vehicle, locate the wires that you will need to connect the unit (refer Fig. 2). Trace the door switch wire that turns the dome or courtesy lamp 'on' and any accessory wire that is powered by the battery when the ignition switch is turned on.

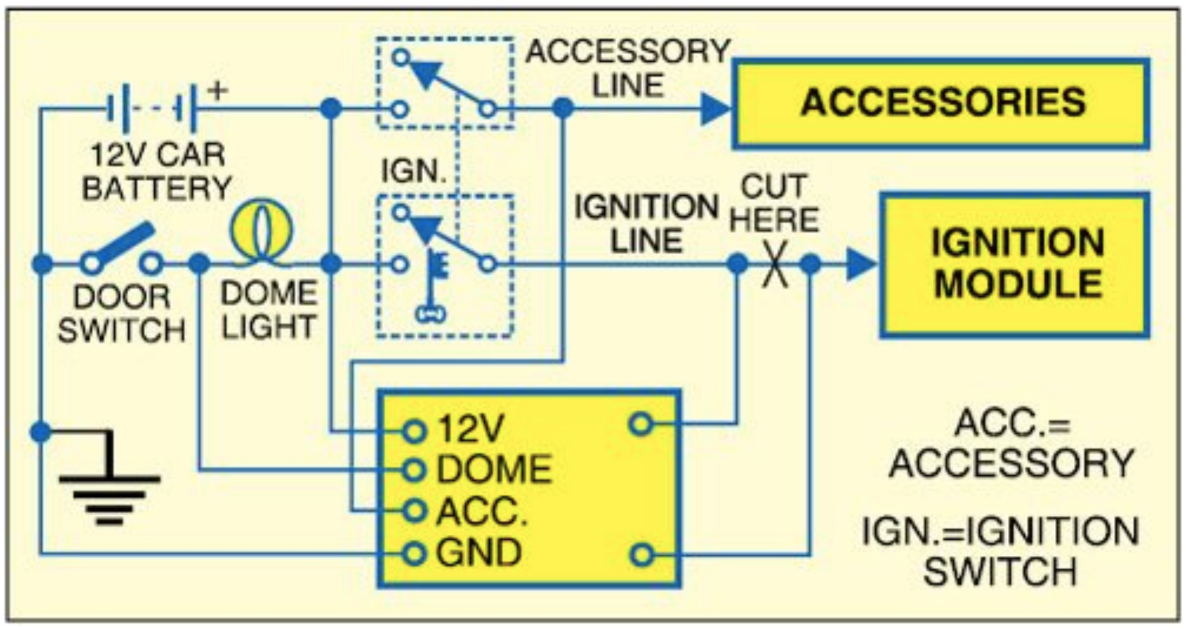


Fig. 2: Wiring diagram

After locating these wires, use multimeter to verify the voltage levels at each terminals. Door switch wire should be at 12V when the door is closed and at zero volt when the door is open. The accessory wire is at 12V only when the ignition switch is turned on. The ignition coil works only when the ignition switch is 'on.' After verifying the proper voltage levels, disconnect the negative (chassis ground) wire from the car battery. This will ensure that any inadvertent short circuit occurring during installation will not cause any damage. Connect the module with dome light and accessory wire.

Secure the anti-carjack module at the desired location, then mount the LEDs and reset switch S1 such that the LEDs are visible to the driver, but the reset switch is hidden.

Locate the power line that goes to the ignition coil section. Cut the line as shown in the wiring diagram (Fig. 2) and connect with N/C contact and pole of relay RL1. Finally, connect ground lead of the circuit to any metal part of the car. After making all the connections, reconnect the negative (chassis ground) battery lead securely. This completes installation of the anti-carjack system.

Final test

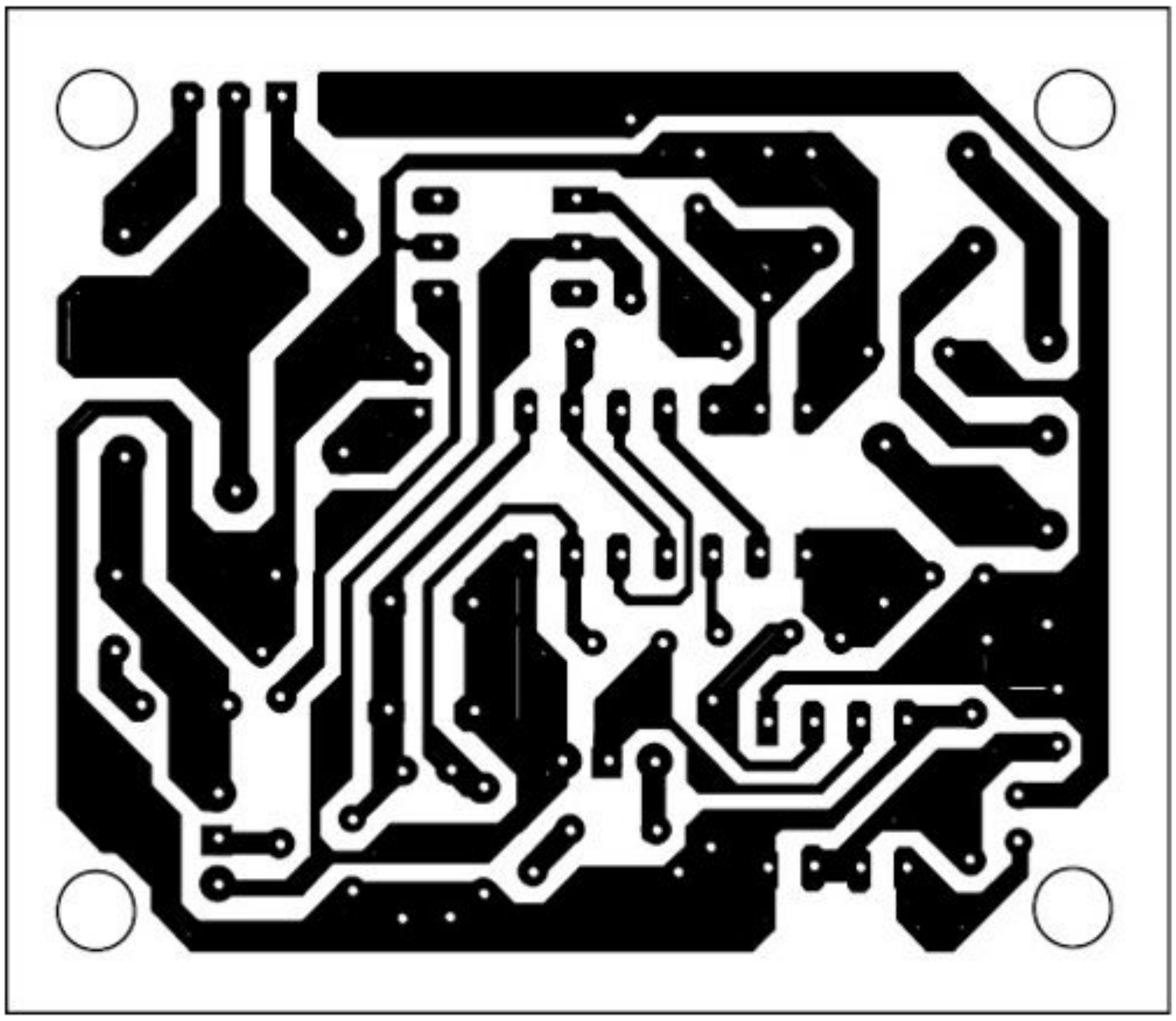


Fig. 3: A single-side PCB for the anti-carjack system

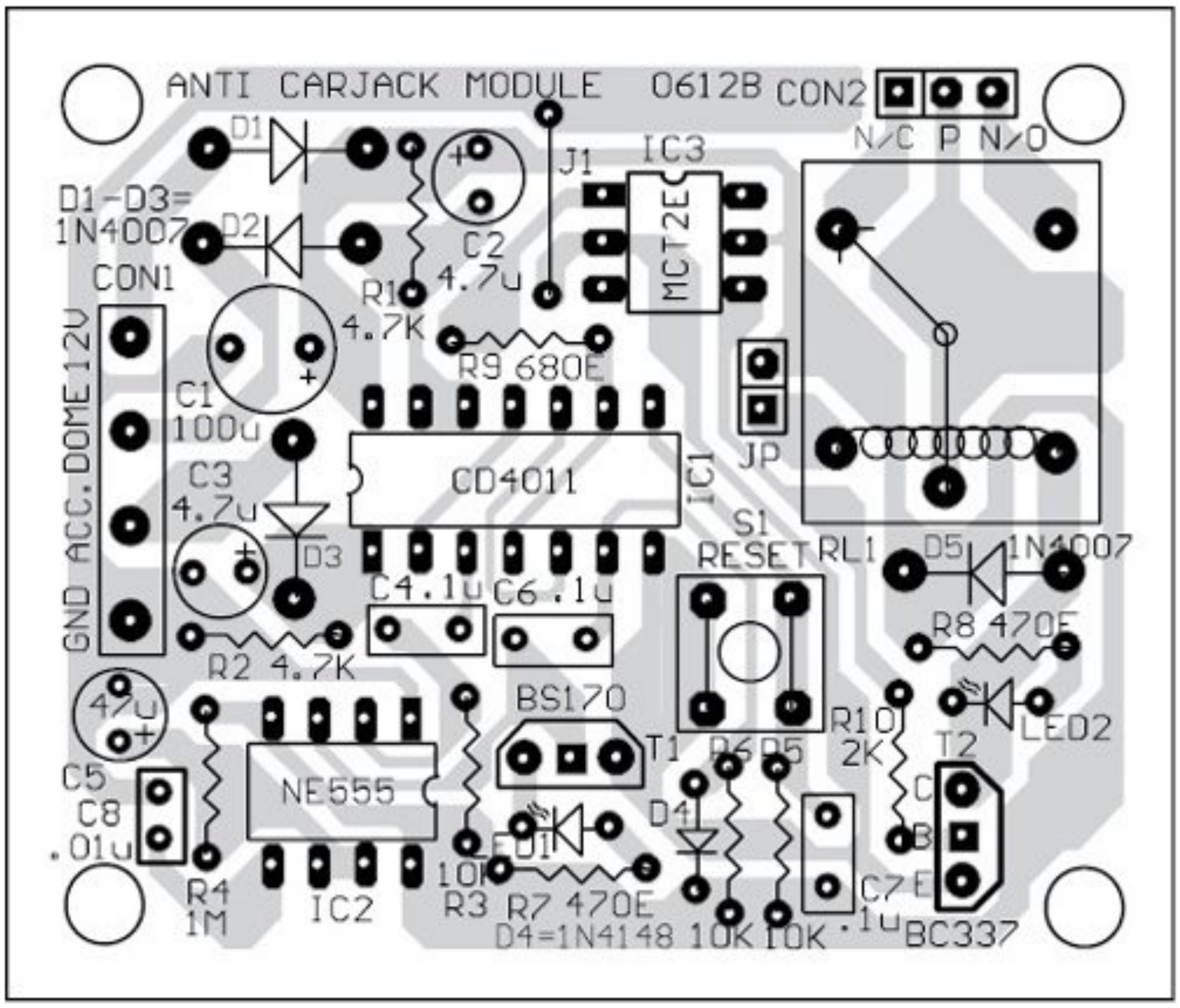


Fig. 4: Component layout for the PCB

When the installation is complete, make sure that both the LEDs are 'off.' If LED2 glows, press reset switch S1 to put it off. Now start the vehicle. When the engine is running, open the door and then close it. LED1 should glow. After a minute, it should go off and LED2 should glow. The vehicle engine will stall and you won't be able to restart it. Press reset switch S1 and verify that both the LEDs are 'off.' Now you should be able to restart the vehicle.

Download PCB and component layout PDFs: click here

The anti-carjack module is always on-duty, fully automatic and never in need of attention. Always remember to turn off the ignition switch before leaving the vehicle. If at any time, the circuit is accidentally triggered as indicated by LED1, wait for LED2 to glow and then press reset switch S1 to put it off. The vehicle will always protect itself in the event of carjack attempt while the ignition switch is ‘on.’

Further application

You can also use this project to protect your car from theft. If at any time you wish to enable the system before leaving the vehicle, open the door before turning the ignition switch ‘off.’ LED1 will glow to indicate that the circuit has been activated. Wait a minute for LED2 to glow, which indicates that the relay is energised and the vehicle disabled. Remember, you must press reset switch S1 before starting the engine when you return.

47

MAKE A BATTERY-OPERATED PORTABLE TEMPERATURE MEASUREMENT SYSTEM

BY J. ANUJ AND DR K. VAIRAMANI

This project presents a K-type thermocouple-based temperature measurement system using MSP430G2553 microcontroller (MCU). The MCU reads sensor data and updates it on a four-digit, seven-segment display, and also sends it to a smartphone through Bluetooth module. The author's prototype is shown in Fig. 1 and its block diagram in Fig. 2.

Temperature measurement is vital in various industrial and weather-monitoring applications. Resistance temperature detectors (RTDs), thermistors and thermocouples are commonly used as sensors for temperature measurement in industrial applications. K-type thermocouples are widely used in industrial applications as these work well in rugged environmental conditions and in various atmospheres. These can provide temperature measurements in the range of -270°C to 1260°C, and an output of -6.4mV to 54.9mV over maximum temperature range. Thermal response of K-type thermocouples is also fast as compared to RTDs and thermistors.

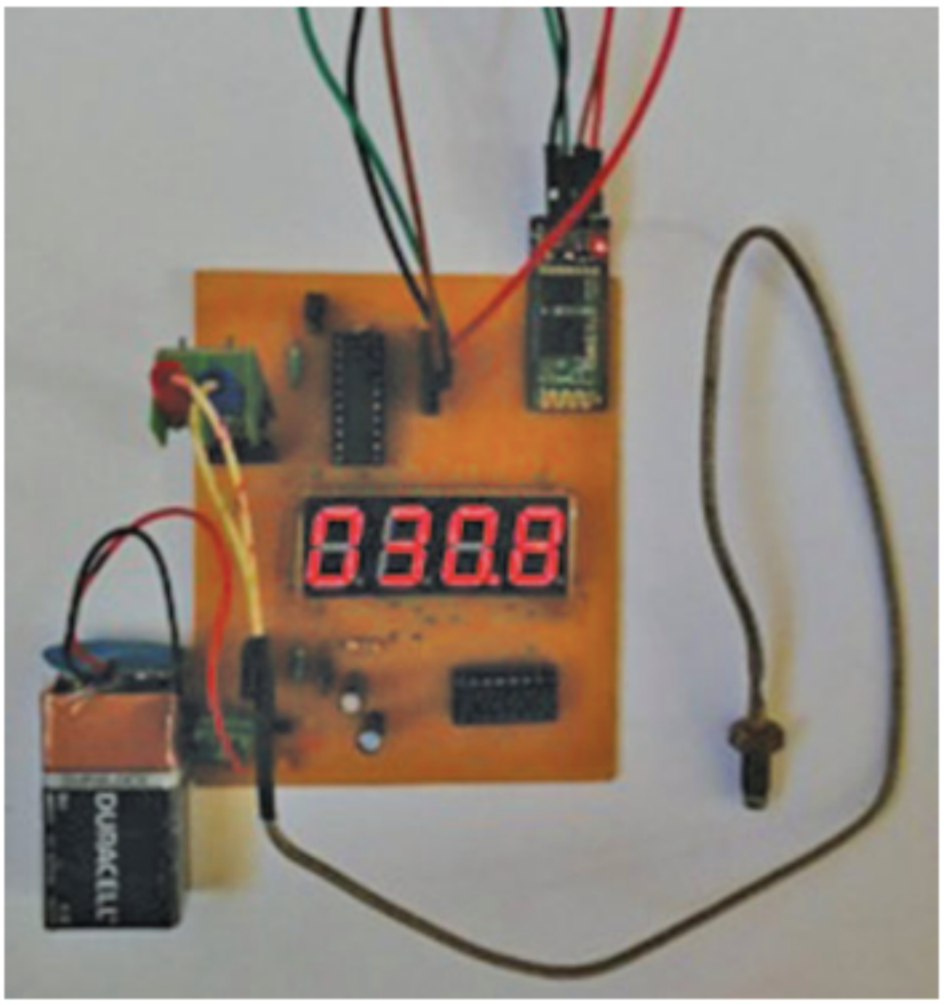


Fig. 1: Author's prototype

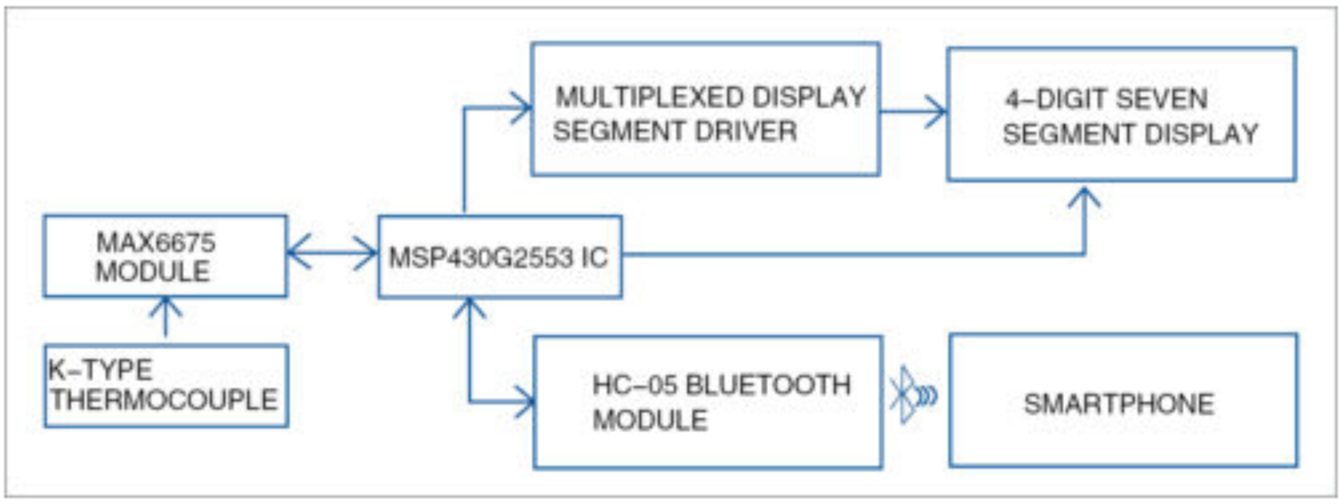


Fig. 2: Block diagram of the temperature measurement system

Circuit and working

Fig. 3 shows the complete circuit diagram of the system. K-type thermocouple is used for temperature sensing and MAX6675 module is used as signal converter. MAX6675 performs cold-junction compensation and digitises the signal from K-type thermocouple. Data is output in a 12-bit resolution and communicated to the MCU

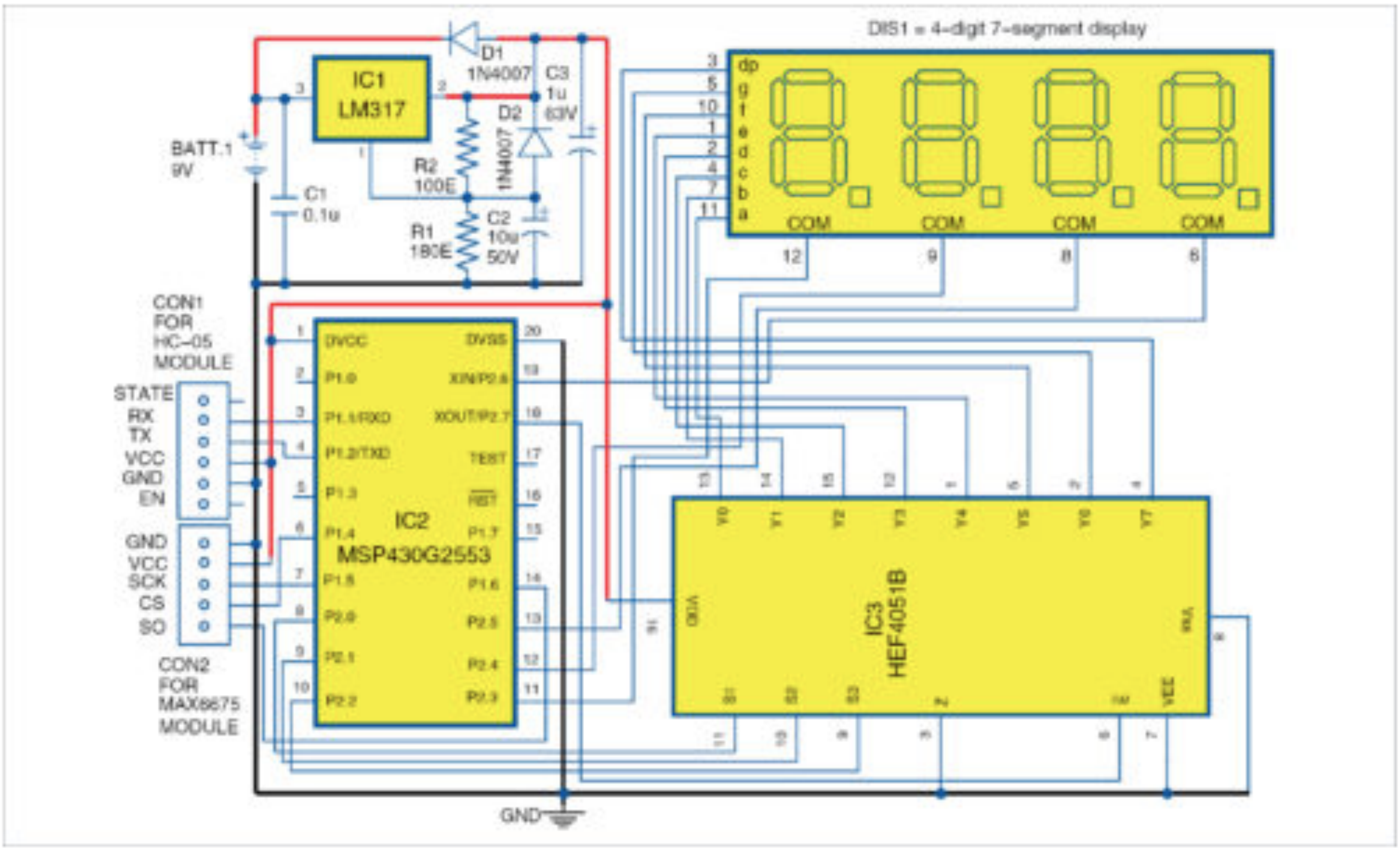


Fig. 3: Circuit diagram of the temperature measurement system

through SPI protocol. MAX6675 provides a resolution of 0.25°C, and allows temperature readings from 0°C to +1024°C.

MSP430G2553 (IC2) is an ultra-low-power mixed-signal MCU with 16-bit RISC CPU, and has a digitally-controlled oscillator for data processing. A twenty-pin MSP430G2553 MCU is used here. It periodically acquires data from the signal converter and converts the digital data into corresponding temperature value.

Converted values are sent to the four-digit, seven-segment display (DIS1) through an analogue multiplexer/demultiplexer HEF4051B (IC3). Through pin P2.7 of MCU, temperature data is sent serially to pin E of HEF4051B to update the segments of the digit.

Pins P2.0, P2.1 and P2.2 of the MCU are connected to selection pins S1, S2 and S3 of HEF4051, respectively, to select the particular segment of a digit in DIS1. Pins P2.3, P2.4, P2.5 and P2.6 of the MCU are connected to pins 12, 9, 8 and 6 of DIS1, respectively, to select a particular digit.

The MCU also sends data to the smartphone through HC-05 module. HC-05 is a Bluetooth Serial Port Protocol (SPP) module interfaced with the MCU through UART. Pins RX and TX of HC-05 are connected to pins P1.1 and P1.2 of the MCU, respectively. HC-05 Bluetooth module is used to communicate between the smartphone and the circuit. A 9V PP3 battery or a suitable DC power supply adaptor is used to power up the system. LM317 is used as voltage regulator to provide 3.3V to 3.6V.

DIS1 is capable of displaying temperature up to 999.8°C. Temperature measurement can also be viewed through an Android smartphone with Bluetooth Terminal HC-05 app installed. Connect Bluetooth of the smartphone with HC-05 Bluetooth module in the circuit. Default auto-pairing password is 1234.

Next, open the app; you will notice a list of nearby Bluetooth devices. Select HC-05 from the list to connect it.

Send ST ASCII value to HC-05. Whenever UART of the MCU gets the command ST, it will send data to HC-05, as shown in Fig. 4.

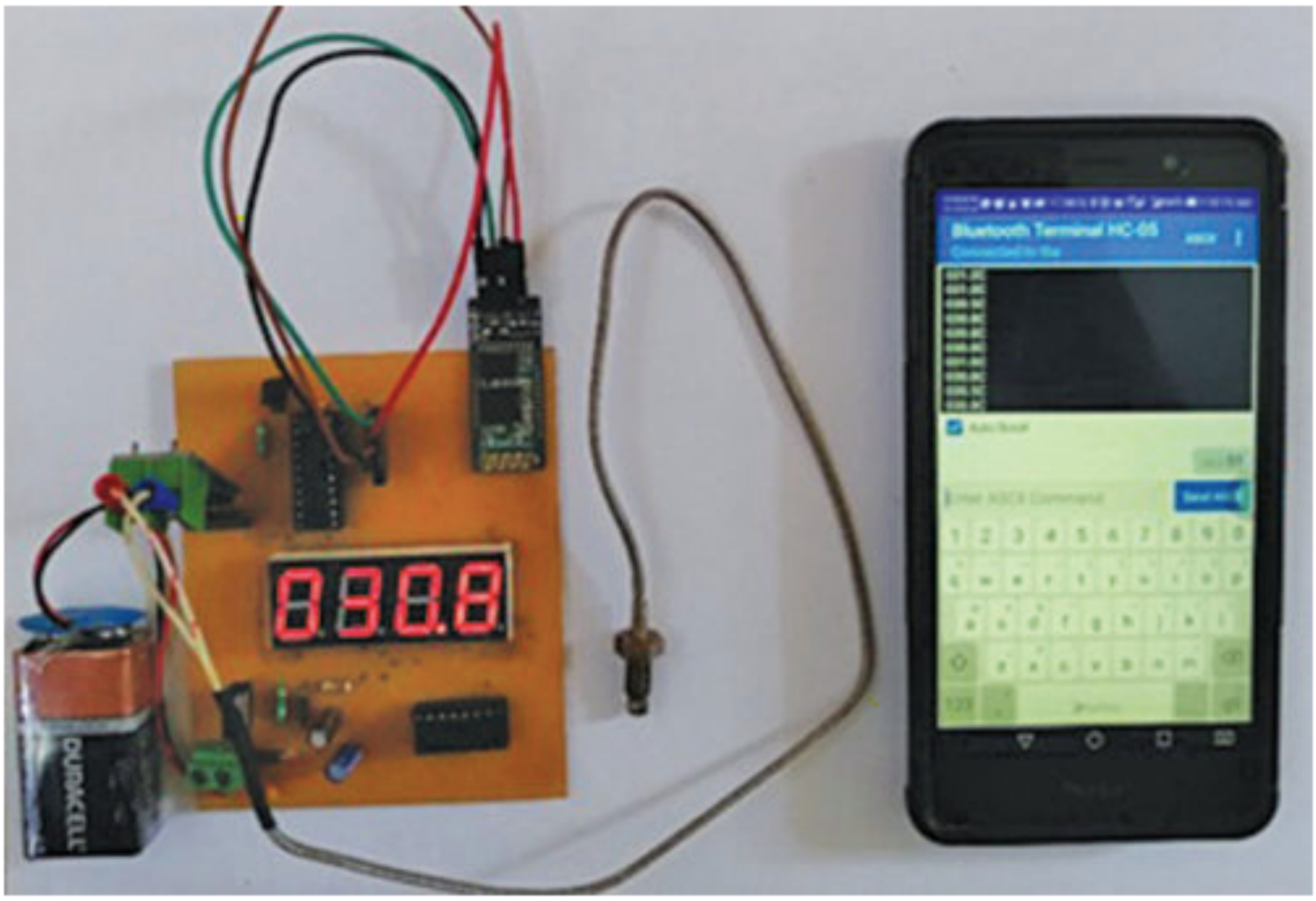
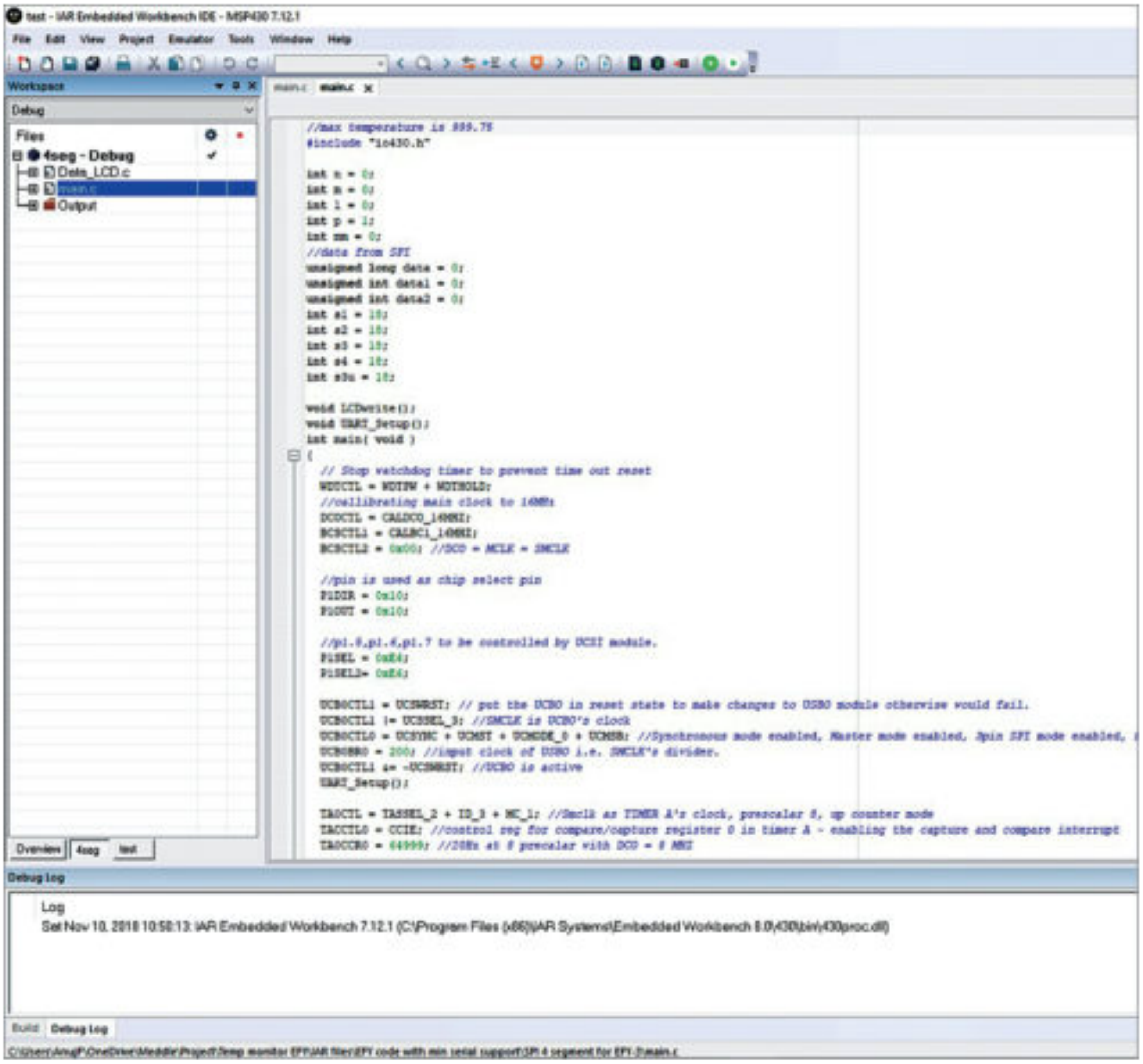


Fig. 4: Output through Bluetooth terminal on smartphone

Software

Fig. 5 shows the integrated development environment (IDE) of the IAR-embedded workbench for MSP430 version 7.12. KickStart edition is used to develop the firmware in embedded C and program MSP430G2553. MSP430 LaunchPad Kit is used to burn the program to the MCU.



```
//max temperature is 999.75
#include "io430.h"

int n = 0;
int m = 0;
int l = 0;
int p = 1;
int mm = 0;
//data from SPI
unsigned long data = 0;
unsigned int data1 = 0;
unsigned int data2 = 0;
int s1 = 10;
int s2 = 10;
int s3 = 10;
int s4 = 10;
int s3u = 10;

void LCDwrite();
void UART_Setup();
int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  //callibrating main clock to 16MHz
  DCOCTL = CALDCO_16MHZ;
  BCSCTL1 = CALBC1_16MHZ;
  BCSCTL2 = 0x00; //DCO = MCLK = SMCLK

  //pin is used as chip select pin
  P1DIR = 0x10;
  P1OUT = 0x10;

  //p1.5,p1.6,p1.7 to be controlled by UCSI module.
  P1SEL = 0xE4;
  P1SEL2= 0xE4;

  UCB0CTL1 = UCSWRST; // put the UCB0 in reset state to make changes to USB0 module otherwise would fail.
  UCB0CTL1 |= UCSSEL_3; //SMCLK is UCB0's clock
  UCB0CTL0 = UCSYNC + UCMST + UCMODE_0 + UCMSB; //Synchronous mode enabled, Master mode enabled, 3pin SPI mode enabled,
  UCB0BR0 = 200; //input clock of USB0 i.e. SMCLK's divider.
  UCB0CTL1 &= ~UCSWRST; //UCB0 is active
  UART_Setup();

  TA0CTL = TASSEL_2 + ID_3 + MC_1; //Smclk as TIMER A's clock, prescalar 8, up counter mode
  TA0CCTL0 = CCIE; //control reg for compare/capture register 0 in timer A - enabling the capture and compare interrupt
  TA0CCR0 = 64999; //20Hz at 8 prescalar with DCO = 8 MHZ
```

```
Log
Sat Nov 10, 2018 10:58:13: IAR Embedded Workbench 7.12.1 (C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.0\430\bin\430proc.dll)
```

[illegible]

Fig. 5: IDE of IAR-embedded workbench

The following steps are used to develop and burn the firmware to the MCU:

1. Install IAR-embedded workbench for MSP430 (IAREW) on Windows PC.
2. Register for KickStart (free) licence.
3. Open IAR-embedded workbench IDE.

4. Select File->New Workspace.

5. Open Project->Create New Project->Expand C Tree Node->Main, Toolchain->MSP430. Click OK.

6. Give a filename and save it.

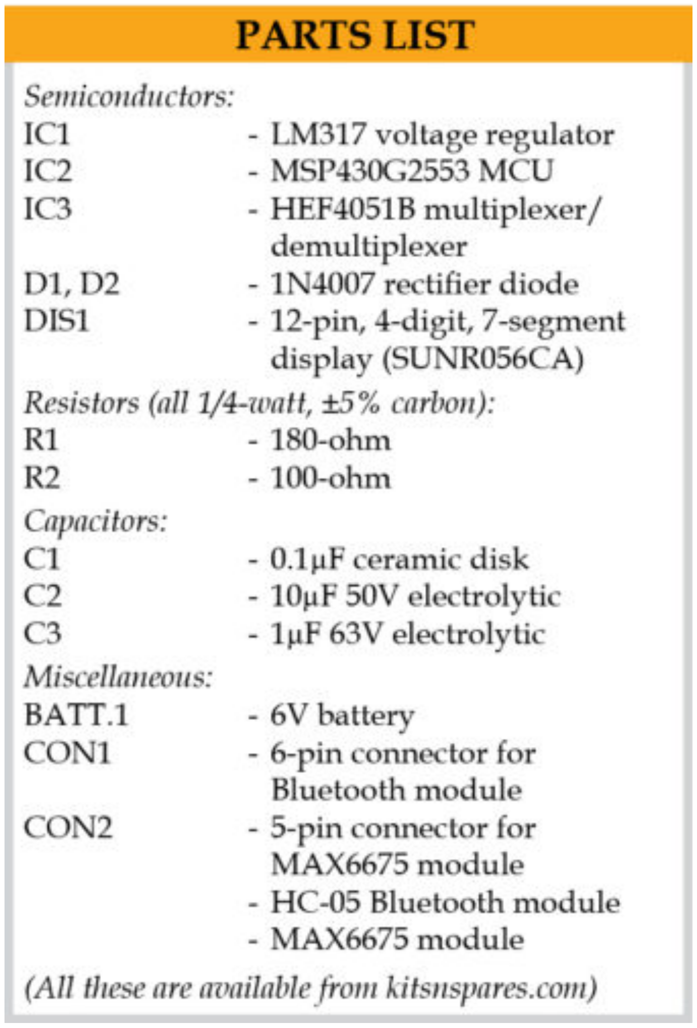
7. Copy and paste the source code (main.c) in the text editor.

8. On the workspace (left pane), right-click on the project name. Select Options->Category->General Options Device->MSP430G2553.

9. Select Category->Debugger->Driver->FET Debugger.

10. Connect MSP430 LaunchPad Kit (with MSP430G2553) with the PC using USB type A male to USB type B mini cable.

11. To burn the code to the MCU, select Project->Download->Download active application.



PARTS LIST

Semiconductors:	
IC1	- LM317 voltage regulator
IC2	- MSP430G2553 MCU
IC3	- HEF4051B multiplexer/ demultiplexer
D1, D2	- 1N4007 rectifier diode
DIS1	- 12-pin, 4-digit, 7-segment display (SUNR056CA)
Resistors (all 1/4-watt, ±5% carbon):	
R1	- 180-ohm
R2	- 100-ohm
Capacitors:	
C1	- 0.1µF ceramic disk
C2	- 10µF 50V electrolytic
C3	- 1µF 63V electrolytic
Miscellaneous:	
BATT.1	- 6V battery
CON1	- 6-pin connector for Bluetooth module
CON2	- 5-pin connector for MAX6675 module
	- HC-05 Bluetooth module
	- MAX6675 module

(All these are available from kitsnspares.com)

After burning is complete, remove the MCU and place it on the temperature measurement target board. It is always good to save the file by giving a filename and clicking on Save.

Troubleshooting

In case there is a problem in running the project, troubleshoot using the steps given below.

1. Download and install MSPDRIVERLIB from this page.

2. Select Category->FET Debugger->Connection->Elprotonic USB-FPA, Manual Selection->Spy-Bi-Wire.

The firmware performs the following functions:

1. Halt watch-dog timer and initialise the internal clock to 16MHz (digital-controlled oscillator).

2. Initialise USCI for SPI and UART.

3. Initialise the timer to create periodic interrupts.

4. For every timer interrupt, read sensor data from MAX6675 and convert that digital data to temperature.

5. Update the value on seven-segment display.

6. If UART receives ST in ASCII, it will keep sending temperature data to HC-05 module for every timer interrupt.

Construction and testing

While constructing the circuit, care must be taken to check all components prior to assembly. This includes checking HEF4051B by connecting digital select pins, ground pin, inhibit pin to 0V and VDD pin to 3.6V.

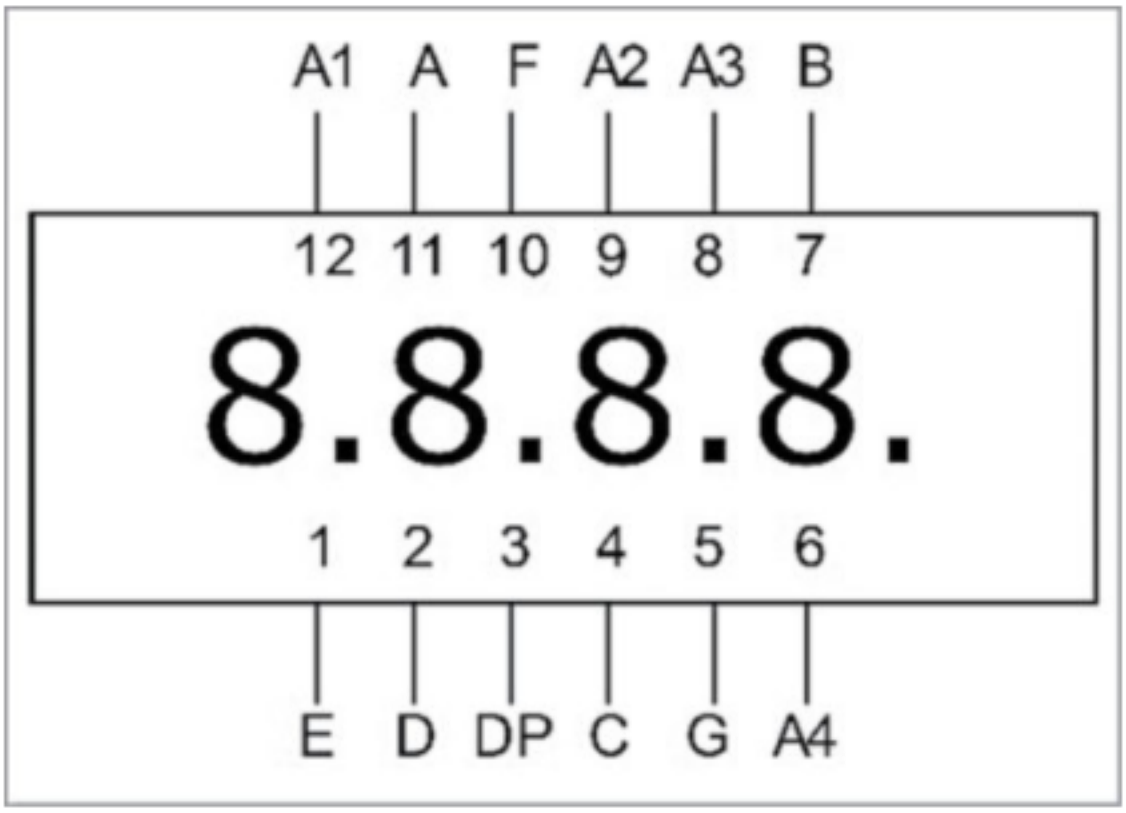


Fig. 6: Pinout of the four-digit common-anode seven-segment display

Now, give input to pin Z; it must be reflected at output pin Y0. If Z is high, then Y0 is high, and vice-versa. Connect inhibit pin (E) of HEF4051B to high (VDD). Output Y0 should be floating.

Test the four-digit, seven-segment display (DIS1) using a multimeter in continuity mode, and map the pins including anode and cathode. Pin details of DIS1 used in the project are shown in Fig. 6.

Need for the resistor to limit current to DIS1 is eliminated due to the presence of internal resistance of HEF4051B.

Whenever the Temperature Measurement System system is powered on, the MCU waits for timer interrupt (~1sec). Upon timer interrupt, the MCU starts to acquire temperature data from the thermocouple sensor through MAX6675 module and converts digital data to valid temperature data, and displays the same on the four-digit, seven-segment display (DIS1).

Download Source Folder

48

RASPBERRY PI TRAFFIC LIGHT USING TENSORFLOW & PYTHON

BY ASHWINI KUMAR SINHA

In metropolitan cities traffic jams are one of the major problems and for the common man and he who use to travel on their legs is one of the major problems. Many time it is seen the people are waiting at crossing to get the traffic clear and the people have to wait and keep standing for an hours to cross the road it become more complicated when any child and old women have to wait for long time to cross the road so today we are going to Raspberry Pi TrafficLight Using TensorFlow & Python that checks how many people are waiting at zebra crossing and from how long they are waiting at zebra crossing and give priority to people rather than vehicles accordingly.

How Our System Works ?

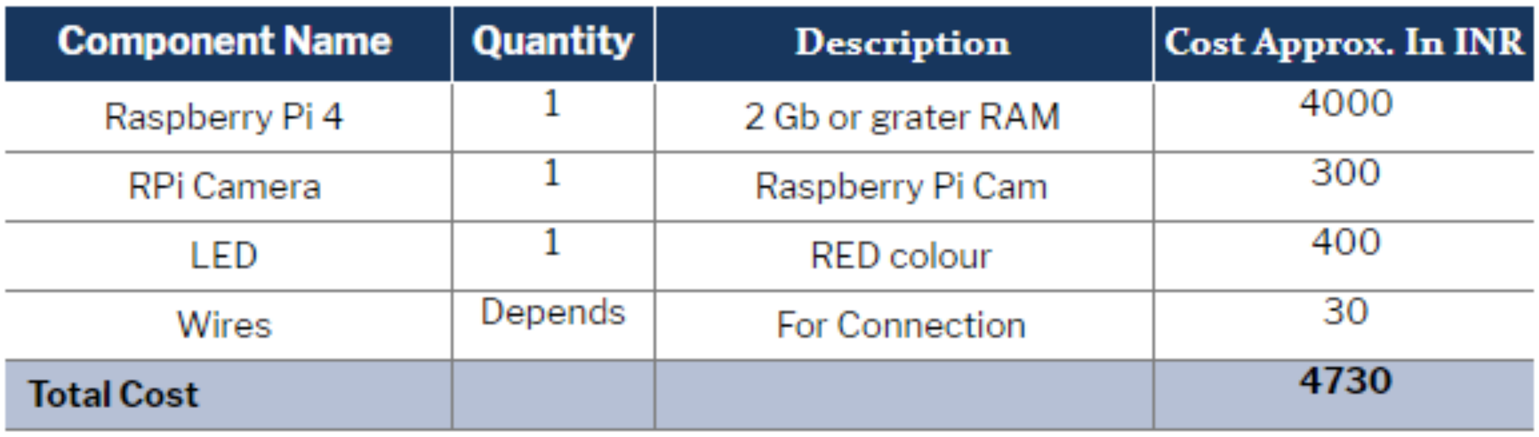
• First a camera streams the live video at zebra crossing

• Then that video is cut in certain frames and a tensorflow with computer vision modules checks the number of people and time from which they are waiting to cross

• If the system get that the number of people is greater than the 3(can be changed) waiting at zebra crossing then it give priority to them. If the number of people at zebra crossing is below 3 but waiting for more than 60 seconds (can be changed) then it give priority to them in crossing.

So let's start our project with some of required components

Bill of material



Component Name	Quantity	Description	Cost Approx. In INR
Raspberry Pi 4	1	2 Gb or grater RAM	4000
RPi Camera	1	Raspberry Pi Cam	300
LED	1	RED colour	400
Wires	Depends	For Connection	30
Total Cost			**4730**

Prerequisites

Assuming that you have already Raspbian os installed and with Python3 environment on raspberry pi and also have access to its desktop.

```
sudo  apt-get update
sudo apt-get upgrade
sudo nano /etc/dphys-swapfile
Then change the line CONF_SWAPSIZE=100 to  CONF_SWAPSIZE=1024
sudo /etc/init.d/dphys-swapfile stop
sudo /etc/init.d/dphys-swapfile start
sudo pip3 install opencv
```

sudo pip3 install numpy
wget https://bootstrap.pypa.io/get-pip.py
pip3 install dlib
pip3 install tensorflow

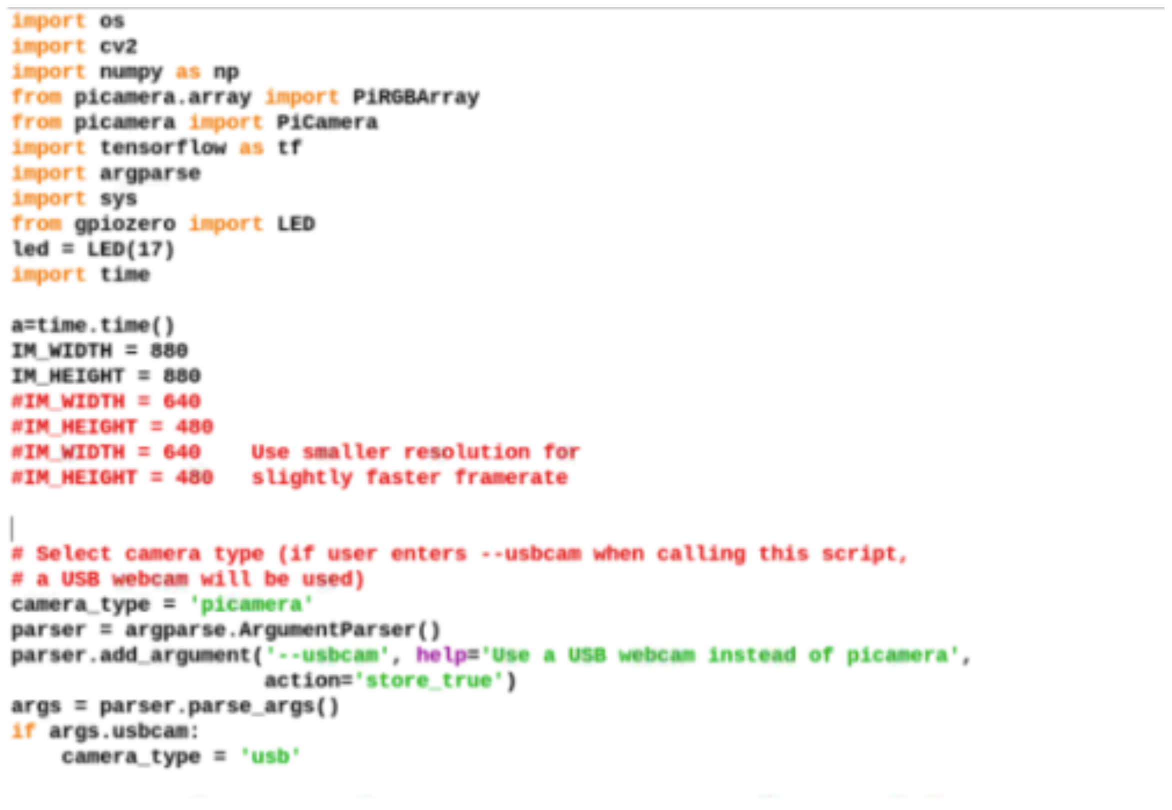
After the installation you can now proceed with the cloning of TF modules, examples, and files using the following command:

git clone https://github.com/tensorflow/tensorflow.git

After successfully cloning, go to the directory → research folder → and paste the code attached with the article. Now open the python3 IDE.Now let's understand and change the code. First we have import the required modules in code and these modules are:

- os
- cv2
- numpy
- tensorflow
- argparse
- sys
- gpiozero
- time

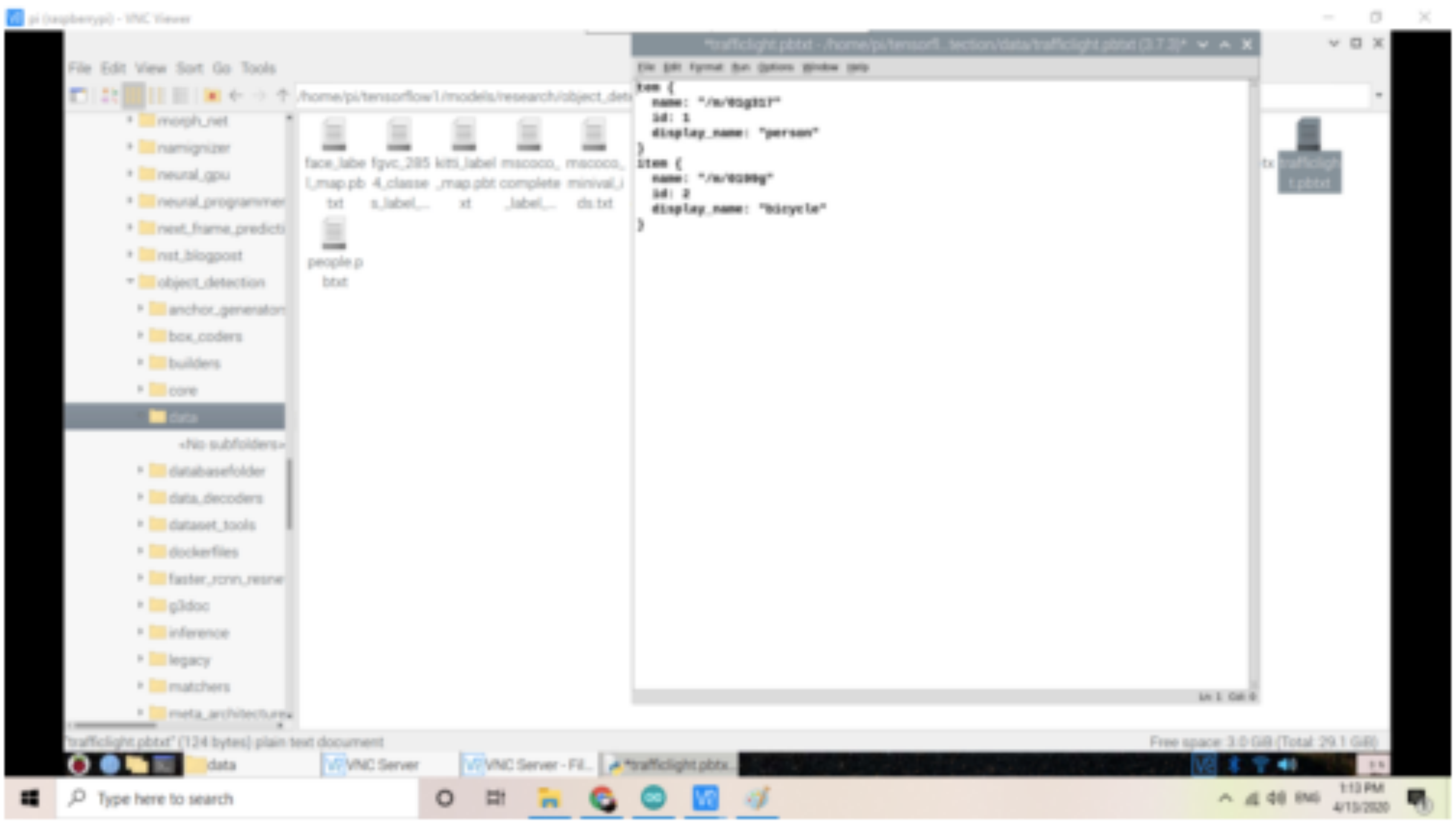
Next we will set the path for tensor flow detection modules and the we will set the name and path for labels here we have using the "trafficlight.pbtxt". Next part of code will check the camera video and cut it in various frames and after that we have code



```
import os
import cv2
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
import tensorflow as tf
import argparse
import sys
from gpiozero import LED
led = LED(17)
import time

a=time.time()
IM_WIDTH = 880
IM_HEIGHT = 880
#IM_WIDTH = 640
#IM_HEIGHT = 480
#IM_WIDTH = 640    Use smaller resolution for
#IM_HEIGHT = 480   slightly faster framerate


# Select camera type (if user enters --usbcam when calling this script,
# a USB webcam will be used)
camera_type = 'picamera'
parser = argparse.ArgumentParser()
parser.add_argument('--usbcam', help='Use a USB webcam instead of picamera',
                    action='store_true')
args = parser.parse_args()
if args.usbcam:
    camera_type = 'usb'
```

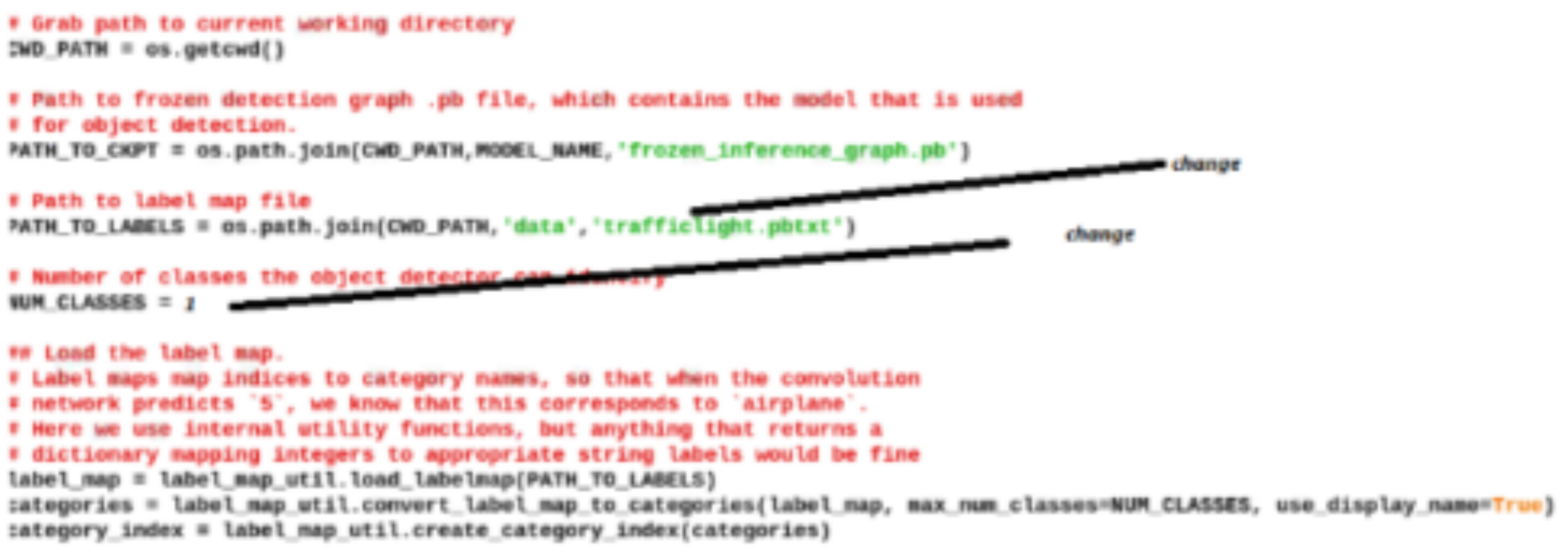
Fig 1

that will try to detect the objects in each frame and then map with the labels that is assigned in traffic.pbtxt . here in traffic.pbtxt have only two labels that is “person” and “bicycle”(you can exclude the bicycle)



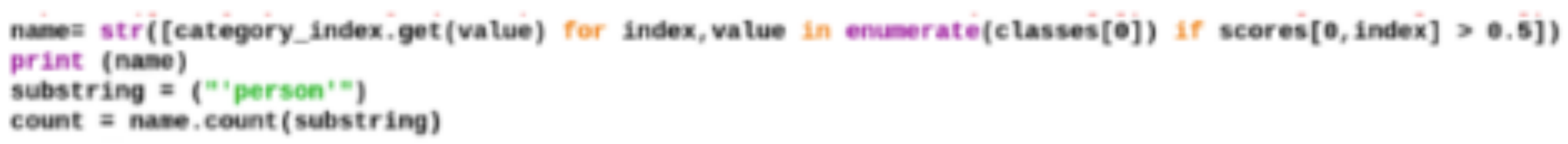
```
tem {
  name: "/m/01g317"
  id: 1
  display_name: "person"
}
item {
  name: "/m/0199g"
  id: 2
  display_name: "bicycle"
}
```

Fig 2.



```
# Grab path to current working directory
CWD_PATH = os.getcwd()

# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,'data','trafficlight.pbtxt')

# Number of classes the object detector can identify
NUM_CLASSES = 1

## Load the label map.
# Label maps map indices to category names, so that when the convolution
# network predicts `5`, we know that this corresponds to `airplane`.
# Here we use internal utility functions, but anything that returns a
# dictionary mapping integers to appropriate string labels would be fine
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)
```

Fig 3

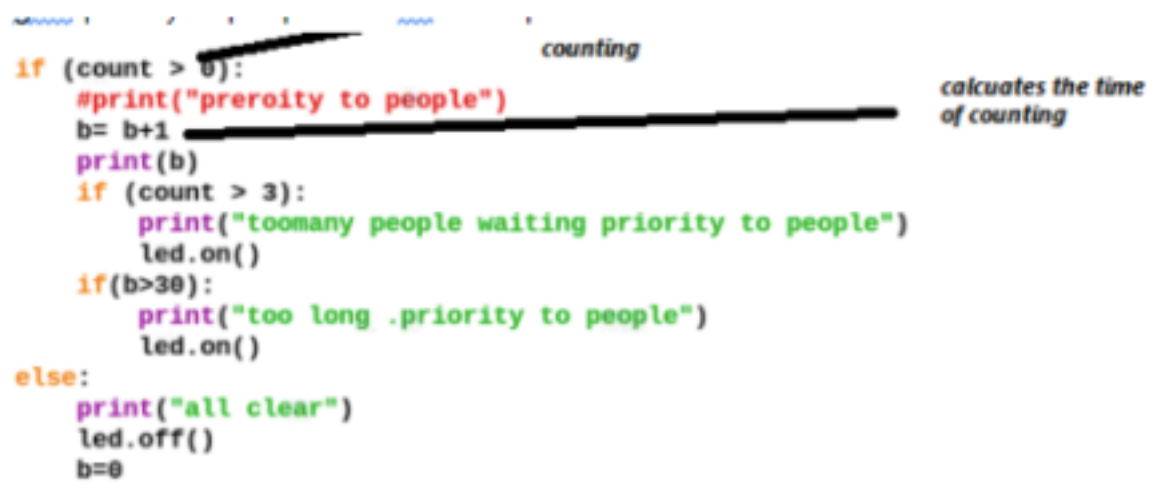


```
name= str([category_index.get(value) for index,value in enumerate(classes[0]) if scores[0,index] > 0.5])
print (name)
substring = ("'person'")
count = name.count(substring)
```

Fig 4.

Now in next part of the code we check the label of object detected and here we have set a substring ie “person” we use this substring to count the number of people in image.

Now we have created several if() conditional statement that checks weather number of people detected is greater than 0 . Next if () statement will check count number is greater than 3 if yes then it turns the redlight on to stop vehicles and let the people pass . if number of people waiting at crossing is less than 3 then it start counting the time and if the time exceeds the 60 min then it give priority to people and let them pass.



```
if (count > 0):
    #print("preroity to people")
    b= b+1
    print(b)
    if (count > 3):
        print("toomany people waiting priority to people")
        led.on()
    if(b>30):
        print("too long .priority to people")
        led.on()
else:
    print("all clear")
    led.off()
    b=0
```

Fig 5.

Connection

Now connect the RPI camera to RPI Camera port using ribbon cable and then connect the components as in diagram (Refer fig 6).

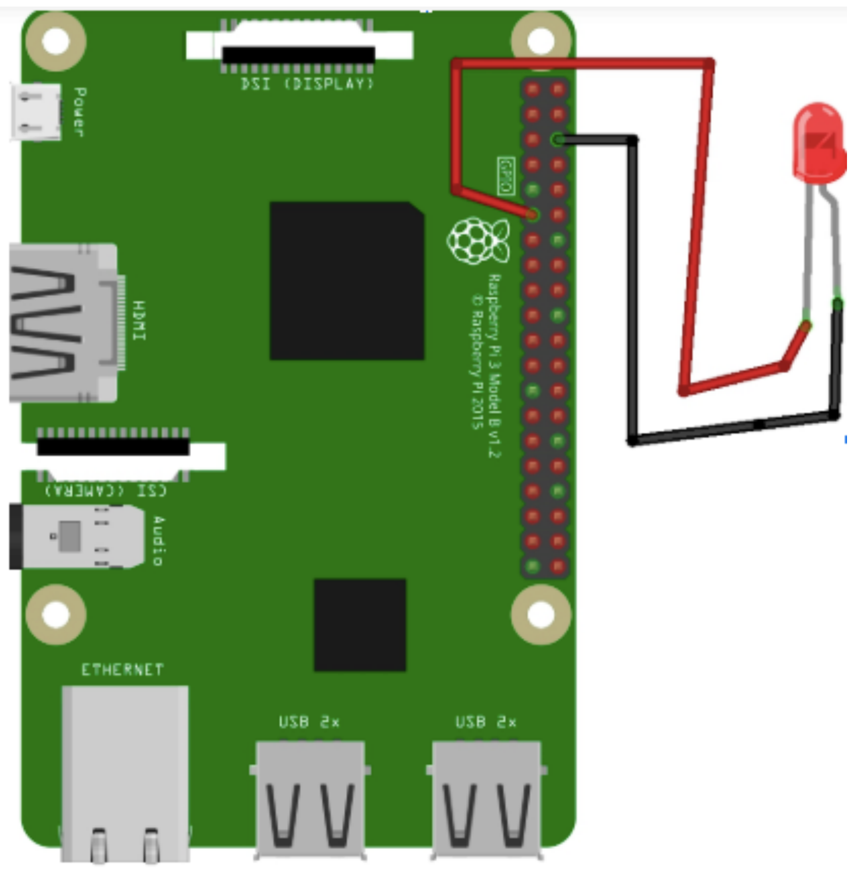


Fig 6.Circuit diagram

Testing

Now save the code and run the

code in python3 IDE and wait for few minutes to load the TF modules and let the camera video window to appear and then bring the camera in front of many people if it detects the number of people detected is greater than 3 then it turn on red LED for stop sign. If people detected is less than 3 than wait for 60 second if the same people is wait for more than 60 seconds then it will let them pass and stop the vehicles by turning the red light on.

Congrats, Raspberry Pi TrafficLight Using TensorFlow is ready!!

Download Source Folder

49

MAKE YOUR OWN TACHOMETER

BY NAJIB GHATTE

A tachometer is an instrument that measures the rotational speed of a shaft or disk in a motor or other machine. Here we present the basic version of the tachometer that shows the revolutions per second (RPS) on a digital display.

Fig. 1 shows the block diagram of the tachometer. Fig. 2 shows the circuit of the tachometer, which is built around timer IC 555 (IC1), IC 7811 (IC2), IC CD4081 (IC3), IC CD4069 (IC4), two CD4033 ICs (IC5 and IC6) and two common-cathode displays (each LTS543) along with some discrete components. IC1 is wired in monostable mode and produces 1-second pulse when triggered by switch S2. Variable resistor VR1 is used for setting the time period of IC1. IC2 generates pulses when the shaft is moving.

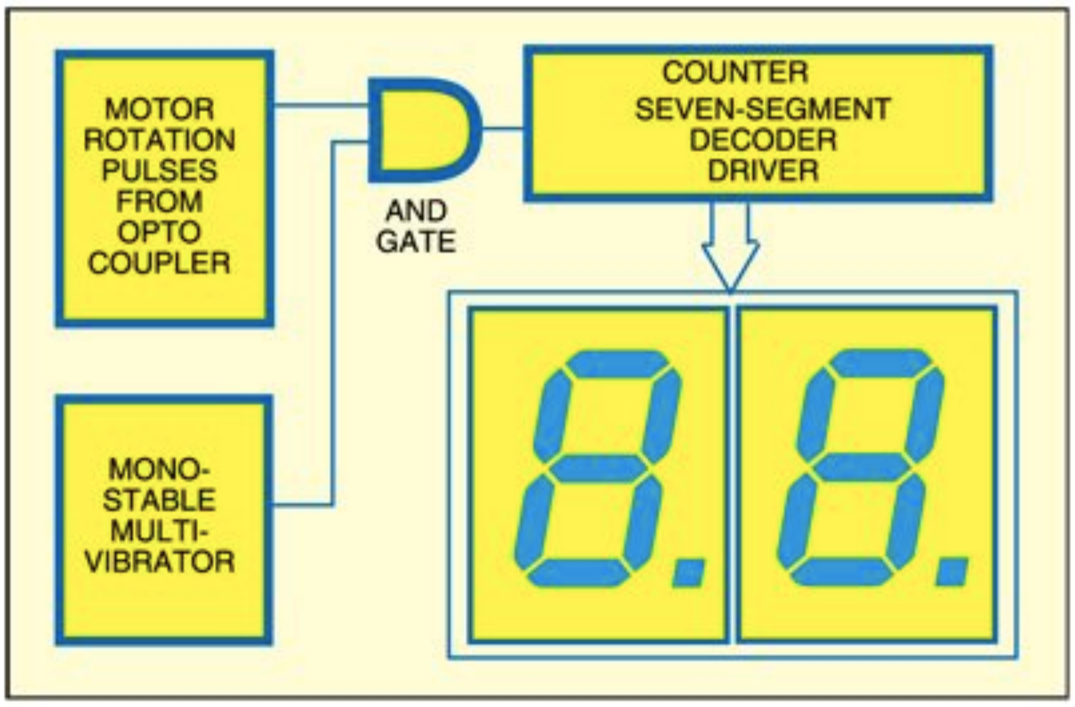

MOTOR ROTATION PULSES FROM OPTO COUPLER
AND GATE
COUNTER SEVEN-SEGMENT DECODER DRIVER
MONO-STABLE MULTI-VIBRATOR

Fig. 1: Block diagram of the tachometer

IC5 (CD4033) is a decade counter with seven-segment decoded output, which is connected to DIS1 for displaying unit's place. Pins 2 and 14 of IC5 are grounded. Pin 5 (CO) of IC5 provides a pulse when the count overflows (count reaches from 0 to 9). This pulse is used as the clock input to IC6 via NOT gate N3. That is, after ten pulses from IC5, there will be an increment of one pulse in IC6. Thus when the unit's counting completes in DIS1, further counting is shown on DIS2. A maximum of 99 digits can be counted and shown on DIS1 and DIS2.

Tachometer circuit

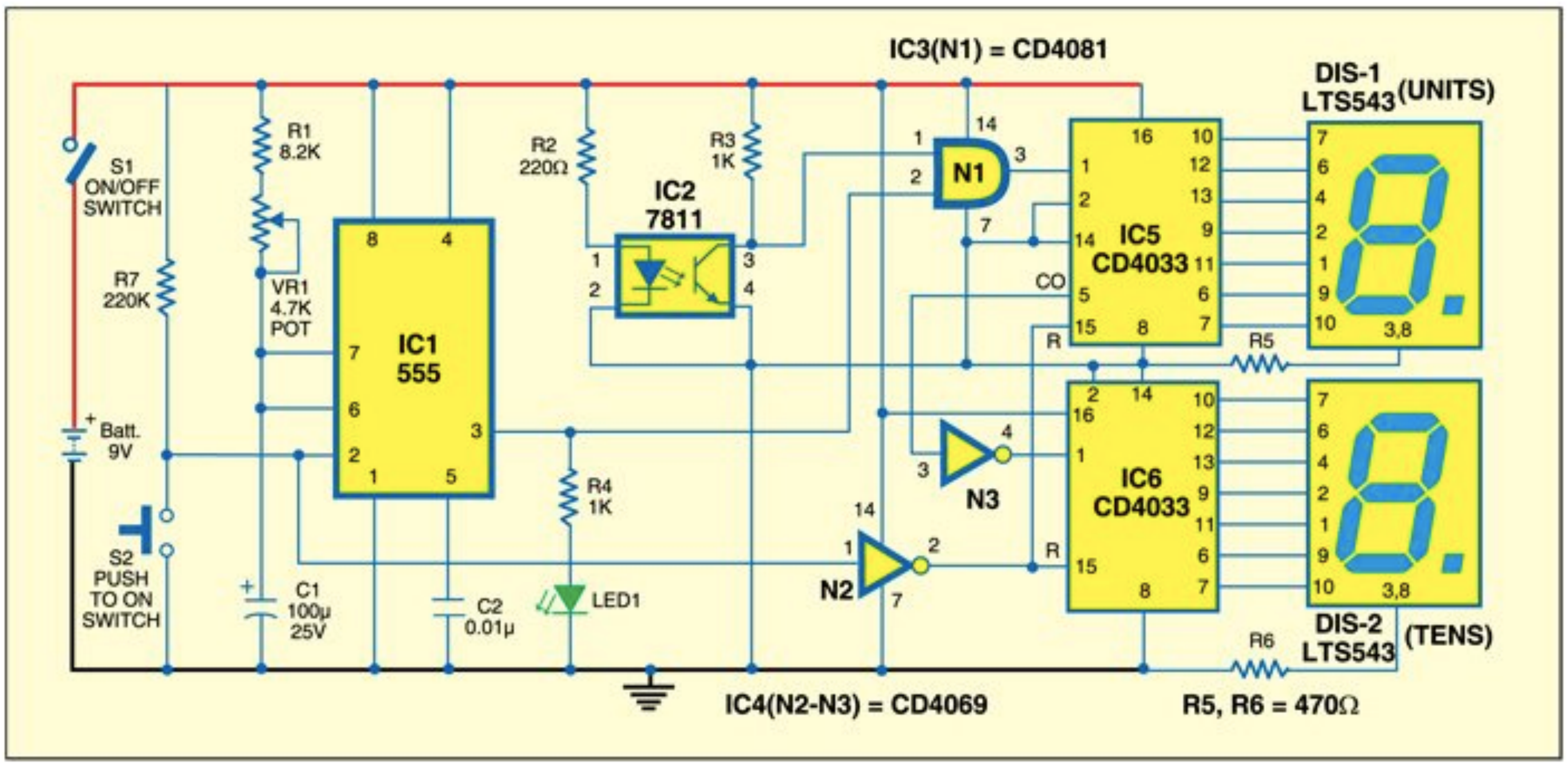


Fig. 2: Circuit of the tachometer

Working of the circuit is simple. Place IC2 such that when the motor shaft rotates, it cuts the IR signal to produce pulses at its output pin 3. That is, as the motor rotates, the shaft cuts the internal light beam within the slot of the optocoupler (IC2). As a result, pulses are generated. This train of pulses is applied to input pin 1 of AND gate N1. Pin 2 of AND gate N1 is connected to output pin 3 of IC1.

To measure the RPS of the shaft, press switch S2 momentarily to enable gate N1 for time duration of one second via the monostable multivibrator (IC1). The pulses are applied as clocks to IC5. The counter increments and generates the code to drive the seven-segment displays. The displays show the number of counts in one second. Accordingly, you can simply calculate revolutions per minute (RPM).

Construction & testing

Assemble the circuit on a general-purpose PCB and enclose in a cabinet with enough space for the battery, switches and motor. Connect the 9V battery and the module is now ready to display the RPS of the motor. As shown in Fig. 3, fix a nose on the motor shaft so that it can cut the internal IR beam of the optocoupler.

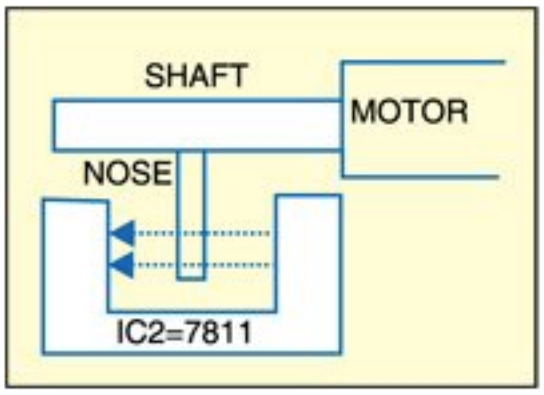
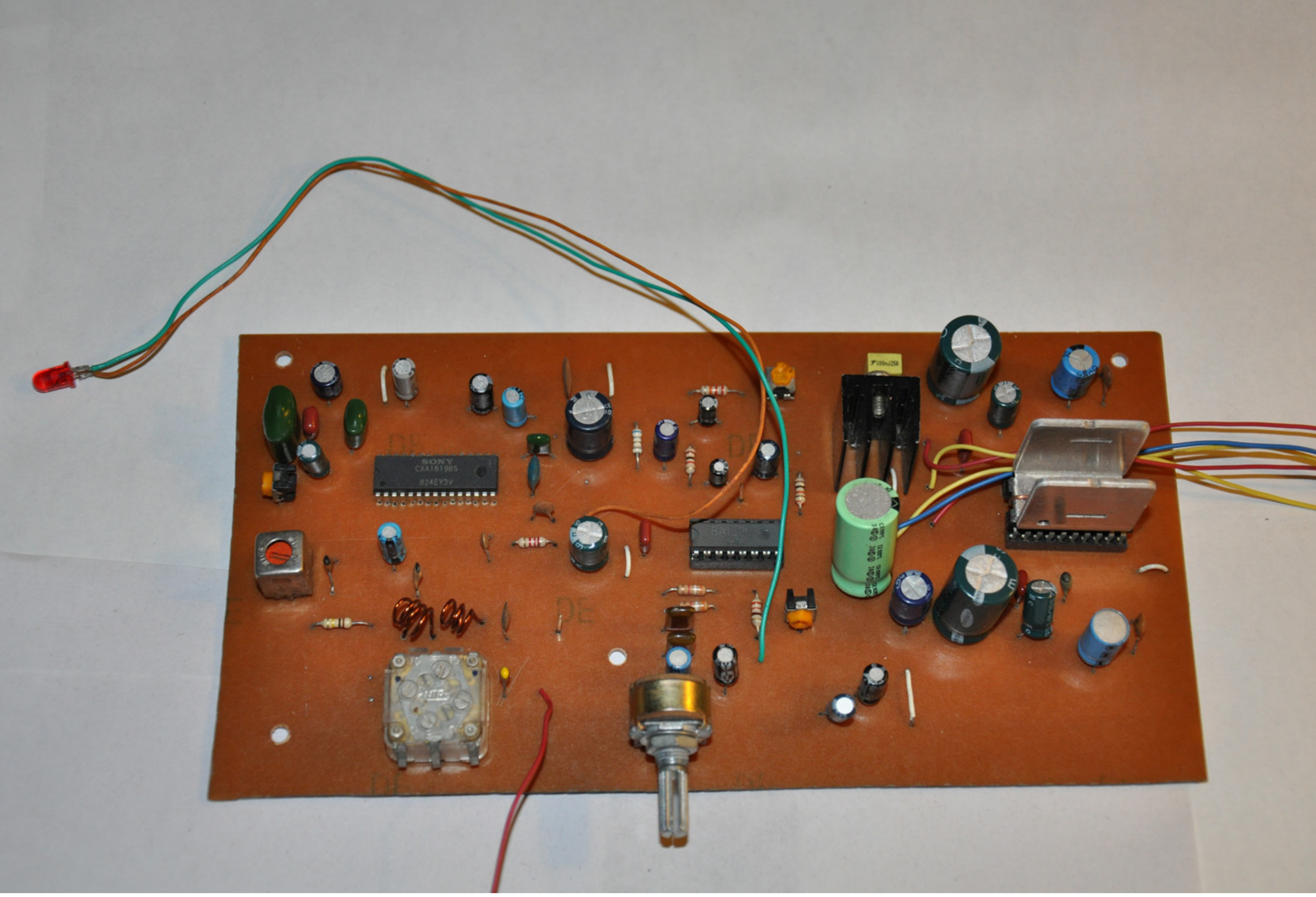


Fig. 3: Shaft setting

The article was originally published in December 2010 and has recently been updated.



50

SIMPLE FM RECEIVER

BY D. PRABAKARAN

A radio or FM receiver is an electronic device that receives radio waves and converts the information carried by them to a usable form. An antenna is used to catch the desired frequency waves. The receiver uses electronic filters to separate the desired radio frequency signal from all the other signals picked up by the antenna, an electronic amplifier to increase the power of the signal for further processing, and finally recovers the desired information through demodulation.

Of the radio waves, FM is the most popular one. Frequency modulation is widely used for FM radio broadcasting. It is also used in telemetry, radar, seismic prospecting, and monitoring newborns for seizures via EEG, two-way radio systems, music synthesis, magnetic tape-recording systems and some video-transmission systems. An advantage of frequency modulation is that it has a larger signal-to-noise ratio and therefore rejects radio frequency interference better than an equal power amplitude modulation (AM) signal.

FM frequency ranges

Frequency modulation is used in a radio broadcast in the 88-108MHz VHF band. This bandwidth range is marked as FM on the band scales of radio receivers, and the devices that are able to receive such signals are called FM receivers.

The FM radio transmitter has a 200kHz wide channel. The maximum audio frequency transmitted in FM is 15 kHz as compared to 4.5 kHz in AM. This allows a much larger range of frequencies to be transferred in FM and thus the quality of FM transmission is significantly higher than of AM transmission. Presented below is an electronics circuit for FM receiver along with its full explanation.

List of Components

- IC- LM386
- T1 BF494
- T2 BF495
- 4 turn 22SWG 4mm dia air core
- C1 220nF
- C2 2.2nF
- C 100nF * 2
- C4 10uF
- C5 10uF (25 V)
- C7 47nF
- C8 220 uF(25 V)
- C9 100 uF (25 V) * 2
- R 10KΩ * 2
- R3 1KΩ
- R4 10Ω

- Variable resistance
- Variable capacitance
- Speaker
- Switch
- Antenna
- Battery

FM Receiver Circuit Explanation

Here's a simple FM receiver with minimum components for local FM reception. Transistor BF495 (T2), together with a 10k resistor (R1), coil L, 22pF variable capacitor (VC), and internal capacitances of transistor BF494 (T1), comprises the Colpitts oscillator.

The resonance frequency of this oscillator is set by trimmer VC to the frequency of the transmitting station that we wish to listen. That is, it has to be tuned between 88 and 108 MHz. The information signal used in the transmitter to perform the modulation is extracted on resistor R1 and fed to the audio amplifier over a 220nF coupling capacitor (C1).

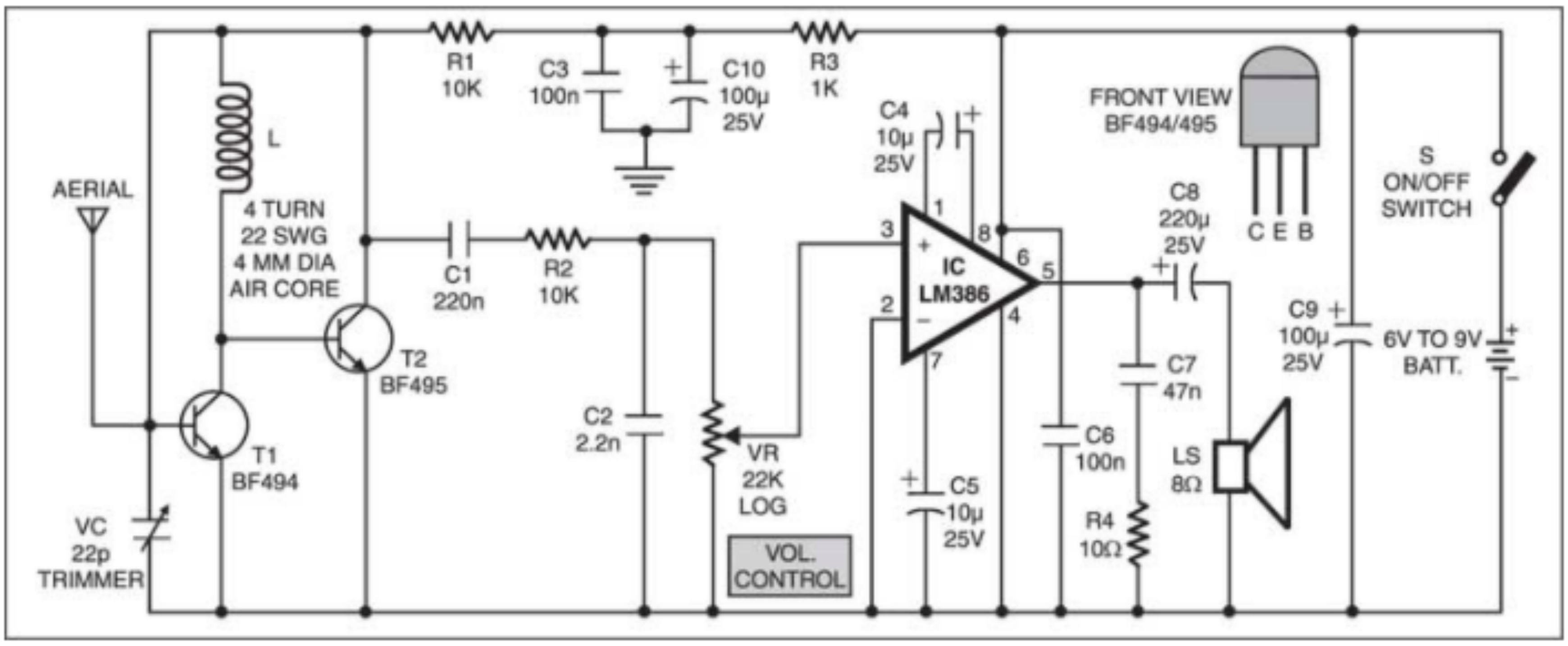


Fig.1: FM Receiver Circuit Diagram

You should be able to change the capacitance of the variable capacitor from a couple of picofarads to about 20 pF. So, a 22pF trimmer is a good choice to be used as VC in the circuit. It is readily available in the market.

If you are using some other capacitor that has a larger capacitance and are unable to receive the full FM bandwidth (88-108 MHz), try changing the value of VC. Its capacitance is to be determined experimentally.

The self-supporting coil L has four turns of 22 SWG enamelled copper wire, with air core having a 4mm internal diameter. It can be constructed on any cylindrical object, such as a pencil or pen, having a diameter of 4 mm. When the required number of turns of the coil has reached, the coil is taken off the cylinder and stretched a little so that the turns don't touch each other.

Capacitors C3 (100nF) and C10 (100μF, 25V), together with R3 (1k), comprise a band-pass filter for very low frequencies, which is used to separate the low-frequency signal from the high-frequency signal in the receiver.

Antenna is a bit tricky

You can use the telescopic antenna of any unused device. However, A good reception can also be obtained with a piece of isolated copper wire about 60 cm long. The optimum length of copper wire can be found experimentally.

The performance of this tiny receiver depends on several factors such as quality and turns of coil L, aerial type, and distance from FM transmitter.

IC LM386 is an audio power amplifier designed for use in low-voltage consumer applications. It provides 1 to 2 watts, which is enough to drive any small-size speaker. The 22k volume control (VR) is a logarithmic potentiometer that is connected to pin 3 and the amplified output is obtained at pin 5 of IC LM386. The receiver can be operated off a 6V-9V battery.

This circuit costs around ₹120.

An EFY GROUP PUBLICATION

EFY ENTERPRISES PVT LTD
D-87/1 Okhla Industrial Area, Phase 1, New Delhi 110020 (India)

www.efy.in